PROTOGENI SECURITY: THREATS TO RESOURCES AND RUN-TIME INTERACTIONS

by

FNU SHALINI

A THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science
in the Department of Computer Science
in the Graduate School of
The University of Alabama

TUSCALOOSA, ALABAMA

2011

ABSTRACT

The ever extending threats to Internet community, and financial, physical, mental, social damages because of it forced researchers to rethink about the architecture, components, and services of future Internet. Security was not the concern at the time of development of existing architecture of Internet There is a very high probability to attack current Internet without being caught which supports the proliferation of cyber crimes. Security is one of the prime objectives of future Internet which is highly obscure term. This is the challenge for ever to maintain the security of the Internet as attackers probably have higher intelligence and determination to break the security.

GENI is a virtual lab to provide all necessary resources and environment closer to expected future Internet so that researchers can test the innovative ideas to develop a more secure, accountable, usable, and manageable future Internet. ProtoGENI is a prototype of GENI and it is in function to test network research ideas. ProtoGENI requires a rigorous observation and implementation improvements to achieve intended security in GENI. Security of ProtoGENI is crucial as experiment results can provide a false picture of security capabilities if they are being tested in an environment which can be manipulated by malicious users or not consistent in its performance. It can affect the security of whole system drastically, and destroying the whole effort of developing a secure future Internet. This work is an effort to test and observe the existing security mechanism and functioning of the ProtoGENI system, and to find out the exploitable attacking loophole. The initial experiments, results, and observations provide a detailed functioning and security problems which can be utilized to improve the overall ProtoGENI security architecture.

Though Security is a process and not a product, this work is to provide the current security issues and suggestions to improve security settings involving all components which work together to utilize ProtoGENI facilities for testing innovative ideas for developing future Internet. Threats to ProtoGENI resources and runtime interactions are in focus for this research work. It explores the existing functioning and possible security weaknesses to cause a non-functional, semi non-functional or malfunctioned system. There are many observations during executing experiments which affect the performance of the system. These observations can assist to improve the overall ProtoGENI functionality.

Results indicate that there are threats to resources and run-time interactions between ProtoGENI components. Non-availability and non-usability of resources can affect the network experiments severely. Cross-experiment communication is also possible in wireless Emulab experiments. Initial Wireless communication analysis on Emulab provides details of wireless traffic behavior and traffic interferences. Overall security at host machine can be enhanced by modifying default security settings including SSH port number and root login rights. An alternative solution is provided to solve default XMLPC server settings to establish initial setup for executing ProtoGENI experiments.

These findings are subject to time-line to the progress of ProtoGENI and GENI projects. This work can assist novice ProtoGENI researchers to understand the basic functionality, associated problems, and possible solutions. These initial findings for security issues in existing ProtoGENI system and observations will assist to improve the overall security functionality of ProtoGENI.

DEDICATION

This thesis is dedicated to everyone who helped me and guided me through the trials and tribulations of creating this manuscript. In particular, my family and close friends who stood by me throughout the time taken to complete this research work.

ACKNOWLEDGMENTS

I am pleased to have this opportunity to thank the many colleagues, friends, and faculty members who have helped me with this research project. I am most indebted to Dr. Yang Xiao, the chairman of this thesis, for sharing his research expertise and wisdom. His feedback was the great source for keep going and trying things. I would also like to thank all other thesis committee members, Dr. Xiaoyan Hong, Dr. Susan Vrbsky, and Dr. Shuhui Li for their support for both the thesis and my academic progress. I would like to thank Dr. Jeffrey Carver for his guidance and motivation for improving information management and presentation. I would like to thank Dr. Randy Smith for allowing me to work in his lab. I am indebted to David Engle and his team to support during several technical difficulties during experiments. I am glad to have support of Ms. Jamie Thompson and Ms. Kathy DeGraw to complete the formal requirements to complete this thesis. I am very thankful to Graduate School for granting financial assistance, needed extensions and for their understanding of my disability during trying times.

This research would not have been possible without the support of my friends and fellow graduate students and of course of my family who never stopped encouraging me to persist. Finally, I thank all of the associated people at University of Alabama, University of Utah, and all GENI-Users for their unconditional support throughout the completion of this thesis.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Computers and Internet are part of everyday life and Modern life around the world is more and more being dependent on internet based applications to learn, to share, to express, to market, to shop, and to help people but unfortunately it is also being exploited as extreme vulnerable source to harass and damage society through various threats. Technology is being improved with every passing year and making life easy and complex simultaneously. It is a constant battle between creators and destroyers. In field of internet, these are frequent questions that what is actually required by world Internet community? Who are main players? Which population is really worried about privacy? Who is suffering from these shortcomings of internet? Does everyone want a secure future internet? Does the world internet user community really understand the threat? It is more like burden on research community who can see clearly the threats, the causes, the consequences, and can judge the possible drastic damages. So the researchers around the world agreed on threats and working on different ideas. There are frequent talks among research groups and discussions. GENI, FIND, FIRE, AKARI, and NICT are few frontiers. There are plenty of information gathering tools to get information from Internet and cost of storing the information is tend to be decreased every year which supports the manipulation of stored information for different purposes. Information stealing or misuse of available information is gone to an extent to warn world internet community to think about security and accountability in internet. There is notable increase in reported cyber crimes every year and the associated financial losses which is only one side of cyber crimes [58] [59].

Internet threats of IP spoofing, Malicious Mobile Code, obfuscated code, throttling limiting the rate of delivery), malicious/tracking cookies, browser hijackers, and distributed DoS (denial of service) provided researchers a cause to work on inducing security (Deterrence, and detection mechanisms, and accountability (Detection, traceability, and universal cyber laws) in internet. These all problems are basically on the exploitation of fundamental flaws in current Internet's architecture which did not consider the security as its primary concern. This basic problem encourages attackers to maximize the attacking effect working on vulnerabilities of software applications and Operating system.

Security needs are being updated in best possible ways with these limitations but unfortunately security cannot be an add-on, so we need to develop a future internet with built-in capabilities to survive the security challenges [3] [4] [6] [7] [11]. Moreover Security is a process and not a product so any security system needs to be capable to detect new possible threats to be pro active to detect any unusual or malicious activity in the network. Detection of any new threat in its budding life can help to develop strategies to tame it before doing much damage to the overall system's functioning. Detection can be very useful in developing resistance against the threats or to have alternative management system to keep the network in good working state in spite of being attacked through some new attacking mechanism which may not be included already into system's surveillance and resistance framework.

"The Internet provides a flexible, scalable, and inter-operable platform that can support many diverse applications. Its popularity has led to a wealth of applications, such as smart phones, PDAs, sensor networks, smart homes, smart grid, emergency and lifeline support, distributed multimedia, and telemedicine."

"However, unrealistic expectations on some of these applications led to the Internet boom in the late 1990s and the burst of the Internet bubble a few years ago." [http://www.computer.org/portal/web/csdl/proceedings/s#2]

It has been acknowledged that future internet application will be more complex, more sensitive and to handle more population of users.

As list drawn from [3], security, availability and resilience, better management, economic viability, longevity, meet society needs, support for tomorrows computing, exploit tomorrows networking, support tomorrows applications, and fit for purpose (it works) are major issues for those current internet has real shortcomings.

It will take time and efforts to test the feasibility of research ideas. Hardware and software resources may not be up to level which is required to do such testing. We need specific arrangements to test research ideas [9] [35].Even we found the research ideas good to implement, total transformation to new infrastructure will be a big challenge. Another challenge will be to draw a universal policy to resolve cyber crimes as different countries follow different set of legal and social rules [1] [29]. A balance of accountability and privacy is a constant question to implement any new security policies on Internet [22] [23] [24] [26] [27] [28]. As switching to future Internet is not possible real soon, researchers are also working to improve current internet [18] [19] [20] until we reach a feasible point to change to new technologies and mechanisms for future Internet.

1. What are the initiatives toward secure future Internet?

Security is one of the fundamental reasons for a clean-slate internet design [3] [6]. Clean-slate approach advocates building future internet with new architecture, new control, management, and security features. The other direction is focused to get as much as possible infusion of security and accountability in current internet.

GENI, FIRE, FP7, AKARI, JGN, NICT, and others came forward to execute research ideas proposed for future Internet [7] [8] [35] [36] [37].

2. What is GENI-Global Environment for Network Innovations?

GENI is a virtual lab [8] for exploring research ideas for future internets at scale, creating major opportunities to understand, innovate and transform global networks and their interactions with society. GENI is supported by NSF and GENI project office is managed by BBN Technologies [25].

During the spiral I of phase of prototyping and designing, awards were made to 29 academic/industrial teams for various projects to build, integrate, and operate early prototypes of the GENI virtual laboratory [30] [31]. Internet2 and National Lambda Rail are partners to build and test prototypes of the GENI system. Cisco, CNRI, Fujitsu, Hewlett-Packard, Infinera, Microsoft Research, NEC, Netronome, SPARTA, and Qwest are working with GENI academic teams across the United States to help build, integrate, and operate early prototypes of GENI [30] [60]. SPARTA [27] is focusing on GENI security architecture.

Spiral II is active. In October 2009, NSF announced two major awards to support an additional 33 academic/industrial prototyping teams, and―for all the prototypes―speeds up federation and shakedown experiments that will guide future GENI system design [30].

NSF is also going to support three collaboration sets of academic/industrial teams to integrate, operate, and host experiments on an end-to-end prototype GENI infrastructure built from GENI-enabled commercial hardware across 14 university campuses, linked by compatible build-outs through two U.S. national research backbones across an aggregate national footprint of 40 gigabits/second [30] [60].

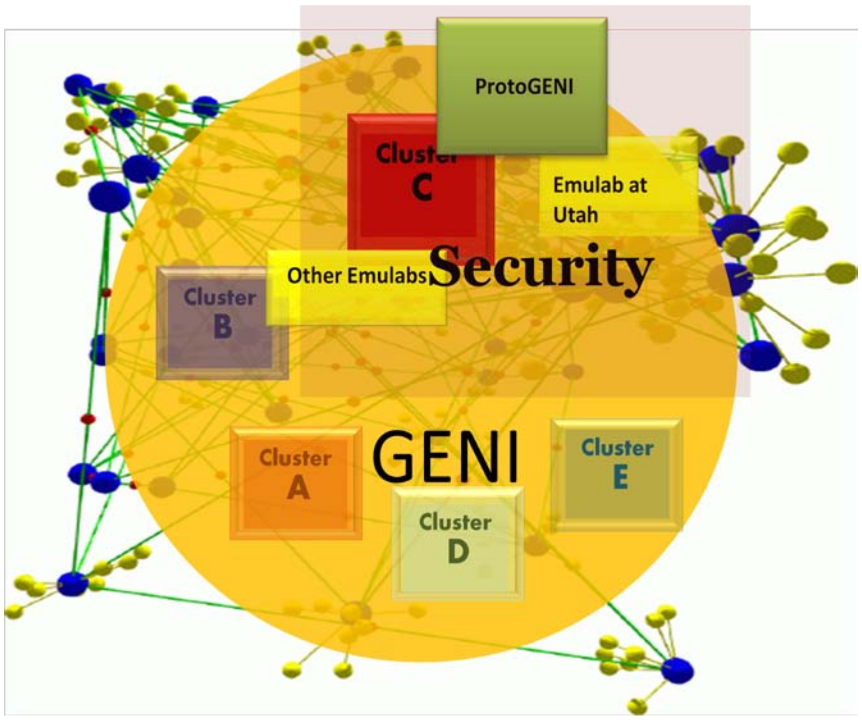


Fig. 1.1: GENI overview including clusters and ProtoGENI

GENI is expected to be dedicated to research experiments but before going for experimenting, GENI infrastructure has to be built as capable enough to provide all necessary resources to deal with complex and advance technologies. It is assumed that GENI is capable of providing facilities to test research ideas for future internet and also capable to resist the attacks against it and to maintain a good working state.

GENI can serve as extraordinary platform to researchers and supposed to be secure and advanced but it is also said that if it could be broken (there is always a possibility), it would be very destructing for the overall GENI functionality.

3. What is ProtoGENI?

ProtoGENI [38] [41] is a prototype of GENI functionality. ProtoGENI is the Control Framework for "Cluster C" of spiral I of the GENI effort [55]. ProtoGENI is based on Emulab [9] software and the slice-based Facility Architecture (SFA).

ProtoGENI includes clearinghouse, slice authority services, and aggregates with associate managers. The ProtoGENI RSpec is the mechanism for advertising, requesting, and describing the resources used by experimenters. RSpecs interact with nearly every piece of the GENI architecture [38] [42]. Emulab [9] was developed by Flux group of Utah University. It is a public facility available to researchers at no cost. An emulated experiment allows you to specify an arbitrary network topology, giving you a *controllable, predictable, and repeatable environment*, including PC nodes on which you have *full "root" access*, running an operating system of your choice. Emulab unifies emulation, 802.11 wireless, software-defined radio, sensor networks, live-internet experimentation and simulation environments under a common user interface, and integrates them into a common framework [9]. An Emulab member can use Emulab resources in ProtoGENI experiments through creation of slices and slivers. ProtoGENI web resources help researchers to know about ProtoGENI [28].

4. Security in GENI/ ProtoGENI

While it is difficult to determine experimentally that a system is secure against all possible attacks within a specific threat model, a number of important security questions can and must be addressed ex. How well does the system perform under different loads?; What are the performance penalties associated with different levels of security achieved in different ways?; Is the secure system, providing a specific level of security and functionality, attractive to users?; How does the experimental system respond to known attacks, carried out in realistic ways?; Can creative attackers, such as researchers from the security research community, interfere with the operation of an experimental system, while it attempts to serve a developing user community? [5] [40]

Any set of “well-behaved” hosts should be able to communicate among themselves as they desire, with high reliability and predictability, and malicious or corrupted nodes should not be able to disrupt this communication [5] [40] [44] [45].

The goal of GENI/ProtoGENI security architecture is to prevent, detect, and manage attacks, so GENI can remain safe and usable by its community of experimental networks and distributed systems researchers. The other goal of GENI is to build security in architecture of globally distributed systems of networks and computers [5] [40]. GENI also believes that it is almost impossible to prevent all possible kind of intentional or unintentional problems, no matter how careful the validation. With all possible threats, GENI believes to have capability to kill any malicious experiment or component which requires adequate monitoring and control management. In the extreme case, GENI is expected to be shut off itself but not every now and then. GENI’s scale, widespread deployment, and visibility make GENI an inviting target for “denial of GENI” attacks [5].

GENI doesn’t claim to provide 100 % technical solution as it has to use many pre-existing components and vulnerable software to be cost effective in early phases [40][44].

5. Security implementation in ProtoGENI

5.1. Authentication and Authorization in ProtoGENI

Currently ProtoGENI is applying one Public Key Infrastructure (PKI) which is covering all ProtoGENI registries, slice authorities, aggregates and principals. ProtoGENI keys pair can be generated on Linux host machine.

ProtoGENI users are allowed to access ProtoGENI resources through Emulab. Each principal has Emulab account and SSL certificate can be generated on user’s Emulab account. This SSL certificate is the identity of researcher which can be used to identify user’s credentials. It allows researchers to activate a session with RPC servers.

PKI is being used to authenticate principals. Credentials are signed by the appropriate authority (slice). X.509 format certificate is being used to implement ProtoGENI public key infrastructure. Certificates are transferred using X.690 ASN.1 DER (Distinguished Encoding Rules) [41]. The aggregate which receives these credentials can verify it using a set of root certificates.

6. Threats to ProtoGENI Components

ProtoGENI [38] is capable to provide its facilities to researchers until all components work together efficiently and adequately. ProtoGENI control framework is Emulab based, and operates on enhanced functionality of Emulab and its subsystems to provide a close system to GENI environment. There are security issues related to different threads which are required to weave an experiment on ProtoGENI.

6.1. The Secure Shell (SSH) and SSL

SSH is a set of programs which employ public/private key technology for authenticating and encrypting sessions between user accounts on distributed hosts on the internet. ProtoGENI allows users only to interact through SSH key. SSH was not designed to protect against password crackers, IP & TCP attacks, and covert channels [50]. SSH-agent remembers the passphrase so user does not need to type it every time while communicating or sending data to the server. This information can be getting through PC backup programs. If assessed, SSH keys are usable by the root user but could not be transported to other machines for indefinite misuse. SSH Keys also can be regenerated if any indication of misuse can be detected.

SSL certification may be forged if it is MD5 algorithm based. It is found that improved algorithm like SHA-1 and SHA-2 are also vulnerable while SHA-3 is under development [50].

6.2. Emulab Security Issues

Emulab does not protect against spoofing on the control network. The test networks are fully separated between experiments by virtualization but virtualization also may have security threats [14]. A port scanner [52] can probe a network host for open ports.

This is often used by administrators to verify security policies of their networks and by attackers to identify running services on a host with the view to compromise it. The threat level caused by a port scan can vary greatly according to the method used to scan, the kind of port scanned, its number, the value of the targeted host and the administrator who monitors the host. The probability of an attack is much higher when the port scan is associated with a vulnerability scan.

A vulnerability scanner is used to assess computers, computer systems, networks or applications for weaknesses. Shared accounts are strictly forbidden on Emulab [9]. SSL and SSH issues are discussed already. Emulab employs a password checking library to ensure the strength of Emulab account passwords. Members of an Emulab project can share files but project details are not readable by members of other projects.

6.3. Other Security issues in ProtoGENI

Unix/Linux Operating system is believed to be more reliable on security measures in comparison to windows operating system and less vulnerable to attacks. Actually, there are threats to Unix operating system like Trojan and other attacks like preventing others from running a program or exhausting system resources [48] [49].

Other than issues discussed above, there is a likelihood of insider attack which can be a result of many possible reasons based on human complex nature [47]. Carelessness in protecting passwords, passphrases, or machine’s unrestricted access to all may also contribute a lot in security breach.

Many parts of ProtoGENI software use the existing software. It is too time consuming and cost expensive to rebuild each and every part of software being used in Existing vulnerabilities in such software may be exploited to threat overall ProtoGENI security [5].

CHAPTER 2

BACKGROUND

Security is one of the strongest motivations for thinking about the developing future Internet. At lower layers of the protocol stack, the current Internet is plagued by undesirable traffic, including spam, DoS attacks, and malicious traffic routed through compromised (zombie) hosts. At higher layers, applications are plagued by various attacks that might be mitigated by security mechanisms at the application layer and lower layers [3]. To date, stopgap measures to fight undesirable traffic via add-on security mechanisms have not been successful. This is not surprising, as the problems stem from two fundamental shortcomings in the design of the Internet: there is no way to reason about the properties of hosts on the edge of the network, thereby assuring the routers of the validity of the traffic emanating from a given host, and there is no way to reason about the properties of services provided by the network, thereby assuring the edge nodes of the integrity of the network fabric. The overarching goal of a new Internet is not just a collection of security mechanisms but an overall architecture for security, which is woven into an overall design for a network [3]. This will require development and experimental evaluation of individual mechanisms, user services, and combinations of components, on a large scale.

Since security means resilience to malicious attack, security studies involve identifying the essential properties that must be preserved in the face of attack, and the *threat model,* which includes the set of actions that an attacker might use to degrade these properties.

While it is difficult to determine experimentally that a system is secure against all possible attacks within a specific threat model, a number of important security questions can and must be addressed experimentally before security improvements can be accepted and adopted in widely used networks. Attention to security, especially in the context of information security, is critical. Priority of access is critical, so past and future work on quality of service (QoS) must be incorporated.

A design for security must be holistic, and deal with issues at all layers. Securing a single layer is insufficient, as an attacker can exploit vulnerabilities in the layers that remain unsecured. To secure BGP, researchers have proposed securing the communication between two BGP speakers by having them share a secret key and adding key-dependent MD5 checksums to each packet exchanged. Unfortunately, this does not secure the semantics of the exchanged information, and a malicious operator (or equivalently, a corrupted router) with appropriate credentials can still send routing messages that disrupt communication system-wide. Even if we secure the semantics of the routing information, the routing protocols can be disrupted by exploiting the vulnerability of the TCP connection carrying the BGP packets to denial of service attacks [3].

A system-wide approach to security is viable, addressing naming, routing, connection management, resource allocation/denial of service, network management, and so forth. While many researchers have begun to tackle these issues, to be practical we need to understand the relationship of these various technologies with each other and collectively on system-wide security. Further, if these technologies have any hope of being widely adopted, we must also demonstrate that they can achieve system-level security at a practical cost

Related work for accountable and secure Internet

Research is going on to build an accountable future Internet with minimum overhead of switching to new infrastructure, new protocols and web policies. There are serious efforts to acknowledge the present and possible future threats to Internet security and reliability, and new concepts are being developed innovatively and with support of previous work done in same problem area. WASCo distributed computing platform provide its net space to run the experiments by untrusted, semi-trusted or trusted clients as PlantLab [12] provides but they do not have a well defined legal relationship between clients and servers. XenoServer project is also building a public infrastructure for wide-area distributed computing which runs multiple virtual machines like PlanetLab node. Grid architectures limit network access to data exchange between participating Grid nodes. The OSU Flow-tool package was also designed to assist in resolving security incidents and complaints. Cisco routers also provide a variety of packages to perform NetFlow data analysis.

In the field of packet loss and delay accountability, studies have been done for Byzantine fault detection in distributed systems; CATS provides accountability for network storage. Some work also has been done on fault localization (FL) tool development, which is similar to the hop-by-hop feedback propagation scheme in loss and delay accountability paper [18]. Trajectory sampling enables all routers to sample the same packets so that ISP can combine the recorded data and can reconstruct its internal path at a router level. This work is taken as basis to builds an alternative Audit system to keep track for loss and delay of packets in network traffic [20]. Similar to AIP [18], self-certifying addressing forms are done. CGA derives the interface portion of an IP address from a public key, and HIP also use hash function to derive a host EID. AIP extends self certification to the whole network.

To implement source accountability, installation of filters at border routers, is the easiest way. AIP’s shut-off packet is inspired by concept given by Shaw [21].

GENI

GENI is a virtual lab [8] for exploring future internets at scale, creating major opportunities to understand, innovate and transform global networks and their interactions with society. GENI is essential, as a platform where a new secure networking architecture, providing strong assurance of communication availability even under attack, can be demonstrated to work in practice on a national scale network connecting millions of users, at Internet speeds and reasonable cost.

The cryptography community continues to explore the theoretical possibilities raised by these and other security-relevant architectural features (such as secure logging, micropayment infrastructure, and source authentication); GENI will provide an exciting opportunity to also compare these features in terms of efficiency, cost, and compatibility with other design considerations for the new Internet [8].

An important thread through many likely security-related experiments is the trade-off between usability and security. Experience with security mechanisms has shown that many ways of strengthening a system against malicious attack make the system less convenient to use. This trade-off can be expected in future systems, since many security mechanisms must distinguish between honest activity, of the sort the system is designed to support, and malicious activity that is intended to disrupt the system.

Although no fundamental theoretical tradeoff has been proved, it generally becomes easier to distinguish honest and malicious activities if honest users take additional steps to distinguish themselves or their actions. Because of the often-observed trade-off, a key goal in security experiments is to evaluate the usability of a system, by representative individuals with no vested interest in the success of the system, in parallel with experiments aimed at determining the resistance of the system to malicious attack.

Privacy is another important goal that requires experimental user communities on a substantial scale. Some forms of security, including any mechanism that makes decisions on the basis of trust, reputation, or authority, will require identity schemes, which must be carefully conceived to balance issues of privacy and freedom from excessive oversight with the goals of accountability [3].

There is a wide range of important and informative experiments that can be conducted on the GENI facility as currently conceived. Examples, some of which are described as follows: Spam-resistant email; Distributed decentralized Access control; Worm propagation and mitigation; Reputation systems; Improved network infrastructure protocols; Selective traceability and privacy; SCADA simulation; Botnet and overlay network security and detect ability; Economic incentives in network infrastructure and applications; Light-weight security tools and algorithms for low-power computing devices; Anonymity in routing and applications; Privacy-preserving data-mining; Secure multi-party communication; Proof-carrying code to protect hosts from malware (and other purposes); Secure electronic cash and micro-payment mechanisms; Experimental combinations of security mechanisms for improved enterprise security [3].

GENI infrastructure includes deep programmability, virtualization, federation, and slice-based experimentation. Different parts of GENI are divided into academic/industrial teams and project development is being managed in spirals.

GENI projects are organized in clusters as Cluster A: TIED; Cluster B: PlanetLab; Cluster C: ProtoGENI; Cluster D: ORCA, and Cluster E: ORBIT. We will describe each briefly GENI Cluster A- TIDE [8]

1) TIED (Trial Integration Environment built on DETER)

TIDE is built upon an architecture (DETER Federation Architecture, DFA) for creating experimental environments across multiple cooperating Emulab-based testbeds. Federator divides the experiment among testbeds as per requirements, constraints and available resources [8] [40].

2) PlanetLab

PlanetLab control framework works on PlanetLab central Software (PLC) and the Slice-based Facility Architecture(SFA). Geni-wrapper module developed to map the GENI specified entities and functions to PLC. As per GENI architecture, PlanetLab defines a clearinghouse, aggregates and slice manager functions. RSpec is used to describe substrate resources [40].

3) ProtoGENI

ProtoGENI [8][28][40] is based on Emulab software and the slice-based Facility Architecture (SFA). ProtoGENI cluster contains several other projects such as DTunnels, CMULab and others. ProtoGENI includes clearinghouse, slice authority services, and aggregates with associate managers. ProtoGENI is using RSpec with descriptions for nodes, links, interfaces, and some metadata.

4) ORCA

ORCA [8] [40] CF working on Java-based resource leasing toolkit called Shirako. It serves as Java reference implementation of GENI clearinghouse, aggregate managers, and experiment control tools. All involved entities exchange digitally signed SOAP messages.

5) ORBIT

ORBIT [8] [40] (Open Access Research Testbed for Next-Generation Wireless Networks) is a radio grid testbed for reproducible evaluation of wireless network protocols.

Other research facilities

European control Framework [2] research includes FEDERICA, Onelab2, Panlab/PII, G-Lab. FEDERICA project deploys an e-Infrastructure composed of network resources for Future Internet experiments. MANTICORE tool and IaaS framework software are making use of web services for provisioning different resources in FEDERICA [2] [4].

Onelab2 operates and built upon PlanetLab Europe (PLE), which provides an open federated laboratory. Onelab2 works on provisioning of control and monitoring functions to support the key needs of experimental research: controllability, repeatability and the ability to share results. It is following PlanetLab but wireless resources are also in consideration.

G-Lab project is a Germany-wide Future Internet research and experimental facility. It is initiated late in 2008 and testing facilities consist of wired and wireless hardware with over 170 nodes, fully controllable by G-Lab partners [4]. WISEBED is wireless sensor network testbed [35], Japanese Gigabit Network-JGN) [36] are few others to mention as prominent testbeds around the world.

Federated Testbeds

Testbed federation is driving a number of research activities in the field and influences the design decisions for testbed control framework [4]. Federation, combining infrastructural resources and services of more than one independently controlled domain, enhances the utility of testbed significantly.

Individual testbeds may have unique infrastructural resources and configuration properties to conduct a specific set of network experiments so federated testbeds can allow experimenters to conduct a new set of experiments which can utilize the combination of unique characteristics of each testbed involved in federated testbed [4] [8] [12]. Federated testbeds are also important as they provide collaboration between different communities like Internet and Telecommunication people or researchers from US and Europe or any kind of collective effort among world communities to develop future Internet. PlanetLab is already working in Europe as its European version. Emulab and PlanetLab are working together to enhance experimental capacity and types of experiments [4] [8] [12]. Federated testbeds are very positive example of going ahead to achieve a common goal of developing secure and accountable future Internet.

- Experiments violated isolation and accidently taken over by an attacker: GENI has to convince hosting organizations not to put their resources to threat by sharing it on GENI and a prompt action in any such case.
- Experiments reporting errors in other parts of system: A trace-back to origin of packets or signal back to responsible party to fix the problem and to prevent it from recurring.
- Misuse by end user: Basic monitoring and tracing tools

- Theft of an experimenter’s credentials to use GENI: More secured way to access GENI, categorized privileges, users to be more attentive on keeping password strong and secure as far as possible.
- Corruption of host operating system software running on the experimenter’s machine: revoking and replacing the user keys and privileges in such cases. It is most likely avenue for attackers against GENI.
- Corruption of any component of GENI, subsequent attacks on other parts: continuous monitoring, mechanism to shut off and to restore the system to a known good state.
- DoS attacks against the GENI management infrastructure: frequent refresh of privileges and possible control plane to reduce these attacks likely.
- Direct attacks against vulnerabilities in the GENI Management software: Many known software vulnerabilities (buffer overflows etc.); not cost effective to consider all issues; GENI intend to rely on detection, confinement and resetting to a known good state to correct intrusions when they occur.
- Privacy of experimental data and the privacy of management policy. Additionally, information leakage from a running experiment is an open research question.[sep06]

Above discussed threats suggests security requirements in GENI as follows: Explicit trust, least privilege, revocation, auditability, scalability, autonomy, usability, and performance are major requirements in GENI security architecture.

INITIAL REQUIREMENTS OF GENI

GENI is intended to support two general kinds of activities: (1) deploying prototype network systems and applications, and learning from observations of how they behave under real usage, and (2) running controlled experiments to evaluate design, implementation, and engineering choices.

Functional requirements

The concept behind GENI is that it can be used to test multiple ideas and concepts. The goal of GENI is to allow long-running continuous experiments. To support multiple, long running experiments, the designers have based GENI on the concept of slices.

One approach to slices is virtualization, an idea that has been a part of CS research for decades. Virtualization takes a physical resource (e.g. a processor) and creates the illusion that it is multiple processors, identical to the original except that each runs slower. GENI must support strong isolation between slices so that experiments do not interfere with each other [3].

Support for security

The desire to support experiments in enhanced security has several implications. First, the GENI infrastructure itself must be stable and secure to an adequate degree, so that that experiments that claim enhanced security or availability can actually demonstrate these virtues. Second, the mechanisms for isolation among slices must be very robust, so that an experiment that involved an attack on a system in one slice cannot “escape” and attack other experiments. Third, there may be a requirement for specialized security technology, such as hardware-specific unforgeable identity tags, key generators, or physical hardware interfaces for secure management [3].

Support for experimenters: Ease of Use; Observability; Fail-safe

GENI must be secure, so that its resources cannot accidentally or maliciously be used to attack today's Internet. GENI is expected to support that an experiment should run within a "bounding box" that limits what it can do; it must be possible to trace network activity back to the responsible experiment (and experimenter), so that any problems or complaints can be addressed; and should GENI enter a period where activities of some components cannot be adequately monitored or controlled, GENI should restrict those activities by other means to a point where safety can be assured (e.g., by shutting down a slice or bringing GENI as a whole into a safe state).

What is critical in GENI Security?

As per GENI Facility Security (Draft) [5], GENI is a tool for enabling fundamentally new types of research in large-scale networks and distributed systems but also warn that it may be formidable attack platform. A GENI slice is a fully programmable substrate and can be configured to perform a wide range of undesirable actions against both Internet targets and other GENI resources. PlanetLab testbed is most used as most close to GENI architecture and it has already experienced such misuses [5]. There are four factors which make GENI more focused on advanced security features.

- GENI will embody more critical and sensitive resources than PlanetLab so misuse of GENI will be devastating.
- GENI is deeply programmable, providing programmability across every layer dealing with outdoor sensors and wireless nodes and all different kind of hardware so on policy may not be work for all.

- As supposed and claimed by GENI goals, it would be at scale large enough not to be managed by some manual handling as currently being done in PlanetLAb, so it would be better to take care of that issue now, instead for waiting it to happen.
- As most promising and prestigious project by NSF, security is among top issues to protect it from converting a most capable attacking facility by malicious users.

What are goals and principles of GENI Security Architecture?

The goal of GENI security architecture is to prevent, detect, and manage attacks, so GENI can remain safe and usable by its community of experimental networks and distributed systems researchers. GENI believes that it is almost impossible to prevent all possible kind of intentional or unintentional problems, no matter how careful the validation.

GENI's scale, widespread deployment, and visibility make GENI an inviting target for "Denial of GENI" attacks. GENI doesn't claim to provide 100 % technical solution as it has to use many pre-existing components and vulnerable software to be cost effective in early phases.

Threat Model and Security Requirements

Three major categories: attacks by outsiders; accidental misbehavior of network, attackers posing as legitimate GENI experimenter; interferences among experiments. GENI may combine multiple elements so should be able to deal with all tree kind of threats simultaneously.

A more detailed view of possible threats are as follows:

- Unwanted internet or RF traffic: GENI proposes to enforce privileges as per level of expertise and a rapid "kill switch to suspend misbehaving experiments and monitoring RF to stop not adequate experiments.
- Disruption to other experiments: stronger isolation between experiments and monitoring shared resources.

GENI Security concepts and mechanisms

GENI is focusing on to discuss uncommon and prominent issues. GENI will be deployed on best available operating systems as new secure operating system development is outside the scope of GENI. GENI proposes to build strong authentication for access control using Public Key Infrastructure (PKI) and certificate authority. X.509 certificates, local authorization policies, validity of authentication by resource manager, and protection of credentials are major concepts.

Protection of private keys is very sensitive. Encrypted private keys can be accessed during machine backup or by access control gained by misconfigured hosts, and then they be tried to break with offline dictionary attack. Attacker will continue to misuse this key until the components started to doubt and found misbehaving patterns and then stop to trust these misused resources.

A password-enabled private key can be made more secure by sharing control over the private key with a GENI component. If Captured-protected Key (CPK) is safe then the compromise of capture-protection server (CPS) does not enable misuse of the Key. Further, storing the private key solely on a hardware token (e.g. smartcard) can reduce the chances of key compromising. Temper-resistant hardware token are real leaders here. Practically, most ideal way of hardware token is not very feasible to all so GENI advocates one or more CPS and hardware tokens on limited bases like GENI administrators [5].

Auditing is an essential part of GENI security architecture. Auditing and authorization are complementary. GENI works on virtual network and it will allow authorized users to interact on GENI so it is auditable that which nodes are responsible for problems as system has their credentials and monitoring allows verification of network traffic activities.

Signature-based intrusion detection and learning-based anomaly detection systems will not be used extensively for monitoring network traffic in GENI as it is not practical to describe normal behavior for GENI experiments and false alarms will be raised because of wide variety of GENI research experiments.

GENI proposes specification-based intrusion detection and anticipates some declaration from each PI about expected behavior of experiments to monitor the behavior of experiment and it can detect any abnormalities. Though it suits to GENI’s nature but may be not straightforward to define an expected behavior of research experiments. GENI proposes specification-based intrusion detection for network behavior but also suggest other common well known detection methods for other activities like Tripwire tool to detect unexpected modifications of files [5].

Challenges in GENI Security

There is possibility of DoS against GENI control plane and GENI is supposed to return to a previous known safe state when its safe operation cannot be assured. GENI control plane expects to adopt the most suitable and most advanced mechanism to avoid DoS attacks during its deployment. Operational security and privacy issues are still in infancy stage and need more discussion, plan, and consideration of challenges to finalize them. GENI has some legacy components and so there is certain level of threat associated [5]. Hosts participating in GENI have strong views about doing experiments on security and aware about possible risks to shared resources. There are many segments in GENI which may cause security issues like vulnerable software, secured hosts, secured communication etc. Network protocols, algorithms, operating system vulnerabilities can create security problems.

Other than technical issues, limitations due to human errors like guessable passwords and keep them in writing, leaving systems unattended and unlocked may feed attackers.

ProtoGENI

ProtoGENI is a prototype of both the GENI software and deployed hardware. Emulab and PlanetLab are two existing bases of ProtoGENI project [38]. The current state of ProtoGENI is reached up to this level after years of work, money and cooperation among researchers from academia and industry.

ProtoGENI was planned as a three phase project: 1) Development of initial infrastructure by taking together the best components of Emulab and PlanetLab to make it available to researchers.

It may not have all desired or defined functionality of GENI; 2) Interfacing of components and modularity of software; 3) Improvement of components and may be replaced by better or alternate versions.

ProtoGENI Control Framework Structure

ProtoGENI CF structure is based on the “Slice-based Facility Architecture” and it has different components handling different functionality and operations. One part, A Clearinghouse includes registry services for Principal, Slice and Aggregate/Component Records, a set of common Registry Services, and certain specialized Principal, Slice and Component Services as fig. 2.1; second part, Emulab sites provide Slice Authority Services. Each Aggregate includes a ProtoGENI Aggregate Manager and zero, one or many Components. An Emulab site can provide access to hosts and other resources required for experiments.

Principals include Researchers and their associated Slice managers. “GENI Management Core (GMC) specifications”, “GENI Distributed Services”, and “GENI Engineering Guidelines” provide basic guidelines to implement in prototype.

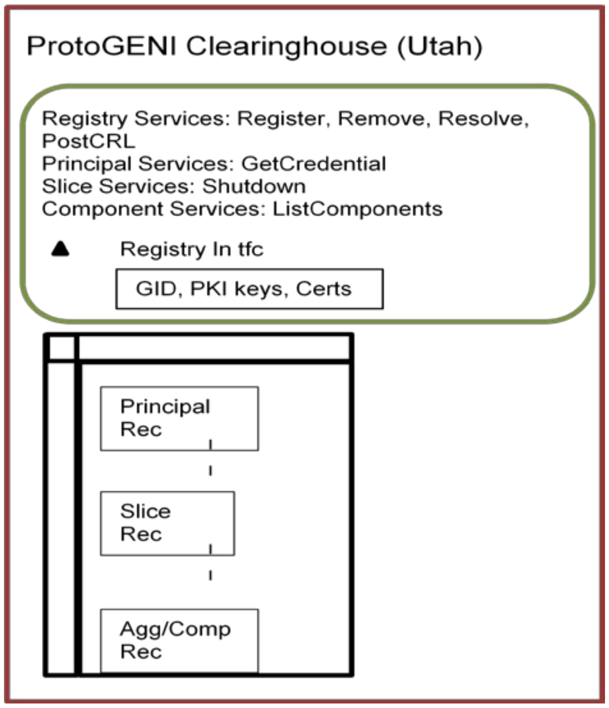

ProtoGENI Clearinghouse (Utah)
Registry Services: Register, Remove, Resolve, PostCRL
Principal Services: GetCredential
Slice Services: Shutdown
Component Services: ListComponents
Registry In tfc
GID, PKI keys, Certs
Principal Rec
Slice Rec
Agg/Comp Rec

Fig. 2.1: ProtoGENI Clearing house [28]

Registry

ProtoGENI implements a centralized clearinghouse and Registry which is co-located with the Emulab site in Utah. The Registry exports a Registry Interface. Principals and Slices are registered in the Clearinghouse by the Slice authority at each Emulab instance. Each registered item has a global identifier (GID). GID contains Universally Unique Identifier (UUID) generated by X.667 open space standard, and a Global Name (GNAME). UUID is never changed, once assigned to an item. The Registry Resolve() and GetCrendential() functions use GNAME or UUID to search relevant identity.

Clearinghouse

Currently Clearinghouse provides registration for Slices, Users, Slice authorities, aggregates, managers and components. It also provides the list of all known aggregate managers. Clearinghouse can shut any experiment or slice.

Clearinghouse operates in registration of new federates. Currently only one instance of clearinghouse is in operation but multiple instances are planned to work together. MySQL database is being used for storage. The location of clearing house is hardwired and most operations are restricted to Slice authority or Component Managers.

To access the clearinghouse, users request credential via XML-RPC. SSL connection is established for existing GID in database. There is no web interface to look into clearinghouse.

Aggregates and Components

A ProtoGENI aggregate may be a cluster node, switch and any other resources running in an Emulab site. An aggregate interface can be exported from an aggregate comprising an Emulab site.

ProtoGENI Principals

ProtoGENI users are basically researchers who are using ProtoGENI resources for network experiments. Each researcher uses a slice manager to register the slice before using the resources so a web client, an XML_PRC client, and an SSH client are also ProtoGENI principals.

ProtoGENI Slices

Researchers can use ProtoGENI slices for their experiments. Slices can be remotely discovered, reserved, configured, and also can be programmed. Researcher can utilize the slice resources as per experiment's requirement. Slices are also basic abstraction for accounting and accountability as it can be known that which user is using how many slices and resources consumed by the slice.

Slice/Sliver: “A slice is a network of computing and communication resources capable of running an experiment or a wide-area network service” [28]. A slice is defined by a set of slivers including a set of network components. A "sliver" can consist of local cluster nodes, including vlan links between them, cluster nodes at different sites.

RSpecs in ProtoGENI: The ProtoGENI RSpec is the mechanism for advertising, requesting, and describing the resources used by experimenters [8] [28]. An experiment can be assembled if the availability of resources is known, available resources can be requested as per requirements, and resources can be allocated or promised to an experimenter [28].

User Management: Users can control the slices once they are bound to a slice at both the Slice Authority and the Component manager. User should get a valid credential to get a slice.

Sliver Interface: When a sliver is created on a component, a sliver interface is also required to give the access to that sliver so that sliver can be used by researchers. A researcher uses SSH login to interact with sliver through sliver interface. Client authentication is provided in SSH login. Setup requires a public key with the sliver server with a matching client private key on user’s local machine [28].

Identification in ProtoGENI

Each registered item on ProtoGENI has a GENI identifier (GID) including Human Readable Name (HRN) and a UUID. The UUID is unique random number generated by X.667 (RFC4122). HRN or UUID is used for lookup operation to execute GetCredential() function [1]. ProtoGENI identifier is SSL certificate that can be generated on Emulab account. A user can regenerate the SSL certificate but same UUID is embedded in it.

Authentication and Authorization in ProtoGENI

Authentication

Currently ProtoGENI is applying one Public Key Infrastructure (PKI) which is covering all ProtoGENI registries, slice authorities, aggregates and principals. ProtoGENI keys pair can be generated on linux host machine. ProtoGENI users are allowed to access ProtoGENI resources through Emulab machines. Each principal has Emulab account and SSL certificate can be generated on user's Emulab account. This SSL certificate is the identity of researcher which can be used to identify user's credentials. Clearinghouse collects all CRLs and sends them to federate as combined single list. If certificate is changed by the user then old certificate is revoked and cannot be used again.

Authorization

PKI is being used to authenticate principals. Credentials are signed by the appropriate authority (slice). X.509 format certificate is being used to implement ProtoGENI public key infrastructure. Certificates are transferred using X.690 ASN.1 DER (Distinguished Encoding Rules). The aggregate which receives these credentials can verify it using a set of root certificates. Nodes may be prevented from being viewed by remote users. Users can be restricted to gain any access to remote nodes. Numbers of nodes can be limited to allot to be accessed by remote users. In a way, ProtoGENI has right to control their resources and aggregates cannot implement their own policies.

Credentials

Credentials are used as identity of user to allow access to remote nodes. Credentials are authenticated documents which describe privileges held by a principal. Privileges to user can be defined in XML document.

A signed credential additionally can include a list of XML signatures of its enclosed ancestors. The expiry time on a child credential may be no later than the expiry time of the parent. There are some privileged and some unprivileged operations in ProtoGENI. The wildcard privilege "*" may be used in place of any other privilege string. Each credential has an owner and a target to claim its right of accessing resources. A sliver is primary object in ProtoGENI and each sliver is controlled through credential which is created when a sliver is instantiated. Sliver duration and resources can be defined in XML document but it is up to ProtoGENI policies to allow access or not. A sliver's resources may be withdrawn after a certain time period if sliver is not renewed by the user. Currently, credential uses XML digital signature format, and XMLSEC library to generate and verify the signed document.

Ticket is a specific kind of credential and it is a variation of credential which includes the description of the resources being promised by an aggregate in the form of RSpec. Credential and ticket both have an expiration date. Currently expiration is not being used in credentials but have a very short time for tickets. Component manager does not respond to expired tickets. To gain resources, user should generate a new ticket but same resources may not be available.

Resource specification (RSpec) describes a component in terms of the resources it has, and also the constraints and dependencies on the allocation of these resources. ProtoGENI RSpec is based on ptop/vtop format designed to assign. RSpec is in "RelaxNG Compact" syntax that is also known as RNC and it is flexible, human readable, and easily can be translated into XML format. ProtoGENI RSpec have details about identification of resources by a UUID, node types, virtualization technology, link details, interfaces, and metadata for generation time and validity time. RSpecs interact with nearly every piece of the GENI architecture [3].

Experiment Setup in ProtoGENI Control Framework

ProtoGENI is a prototype of GENI and expected to provide basic functionality necessary to run experiments by GENI researchers. Two trusted XMLRPC servers implement the Slice Authority API and the Component Manager API. The CM is actually an aggregate component manager. Each user who wants to use the GENI interfaces creates a password protected SSL certificate via the Emulab web interface. Only registered Emulab users at one of the federation sites can use the GENI APIs. You then use your favorite xmlrpc client to talk to the servers. Python is preferred because it's really easy to write a client program. The source code has a bunch of test programs that demonstrate how to do this and use the APIs.

A list of physical resources can be got but logical status is more important to know about the availability of resources. Unavailable resources can be hold if not available currently and can be tried again to get those busy resources by putting reservation or waiting status for those resources.

Component Programming

Researchers are allowed to login on their assigned sliver (component). Booting and load code can be done through sliver interface [28].

Resource to Resource connections

Researchers can apply resources from two aggregates than it is required to connect those aggregates. An RSpec can include a request for a link between a set of nodes. ProtoGENI supports independent control of these links, LANs, and individual interfaces on nodes attached to links.

Secure Shell- SSH and OpenSSH

Secure Shell or SSH is a network protocol that allows data to be exchanged using a secure channel among network users. SSH was designed to replace Telnet and other insecure remote shells, which send information, notably passwords, in plaintext, which make them vulnerable. The encryption used by SSH provides confidentiality and integrity of data. SSH uses public-key cryptography to authenticate the remote computer and allow the remote computer to authenticate the user, if necessary. SSH is very commonly being used by Linux and other Unix like systems. SSH is typically used to log into a remote machine and execute commands, but it also supports tunneling, forwarding TCP ports and X11 connections. SSH can transfer files using the associated SFTP or SCP protocols. SSH uses the client-server model. The standard TCP port 22 is set as default port for SSH services. An SSH client program is typically used for establishing connections to an SSH daemon accepting remote connections for communication.

Protocol 2 is being used by installed version of OpenSSH. For protocol 2, forward security is provided through a Diffie-Hellman key agreement. This key agreement results in a shared session key [13][15]. The rest of the session is encrypted using a symmetric cipher, currently 128-bit AES, Blowfish, 3DES, CAST128, Arcfour, 192-bit AES, or 256-bit AES. The client selects the encryption algorithm to use from those offered by the server. Additionally, session integrity is provided through a cryptographic message authentication code (hmac-sha1 or hmac-md5) [13][15]. MD5 is least significant algorithm among options being applied for encryption. Sha-1 and sha-2 are better but also found some vulnerability against these two also. As for now sha-1 or sha-2 are almost on same level and sha-3 is under development [14] [15].

The server and the client enter an authentication dialog. The client tries to authenticate itself using host-based authentication, public key authentication, challenge-response authentication, or password authentication.

Related Work to SSH Security Issues

SSH was developed to keep network traffic protected through encryption but studies show that in recent years, brute-force attack against SSH, ftp, and telnet servers is most common form of attack [63]. Linux is most popular operating system among today's Unix type systems. Again Unix/Linux system is considered safer in comparison to Windows OS but as per one study, Linux is 'most breached' OS (65% of 154,846 hacked systems) on the net (web 2004). Though it is said that an updated OS with latest patches is secure but brute-force attacks against SSH are possible on fully updated Linux OS. Earlier ProtoGENI [28] resources, Emulab [9] machines used FreeBSD as its default OS but now Linux is the default OS, and there is always a possibility to breach the security because of OS vulnerabilities. Even FreeBSD OS has been reported to vulnerable to certain kind of vulnerability but it is most secured OS at present by reports [62] An experimental study shows that weak and guessable passwords are among other reasons to attack against SSH [61]. Literature suggests that default port 22 for SSH service and other default settings may provide an easier way to attack [65] [66]. Most of the existed attacking kits have inbuilt scripts to exploit these default settings and attacks are almost nil if these default settings are changed to a different non-standard port. Technically it is not strong way to enhance security as it is just a change of port no. but practically it is more work for attacker as port scanners may or may not respond to this non-standard port so it is always hard to guess which non-standard port is listening on host. Disabling logins via SSH for the root account is another way to add strength to SSH security.

System can monitor all failed attempts to login and should block the IP addresses after repeated failed login attempts. Iptables also can be used to restrict the users to the SSH server port. SSH-keys are also preferred in place of passwords to keep system more secure.

Port scanners are commonly used to know network statistics about open ports and associated services. It seems obvious that an attacker will first try to know the possible ways to enter into network and so a port scanner comes handy to do this. Researchers tried to find out the relationship between a port or vulnerability scan and following attacking possibilities, and they also tried to examine behavior of attacker after compromising of SSH to have an idea about severity of levels of attacks through SSH [65]. Results show that only 4% of the port scans were followed by attacks while 21% of the vulnerability scans were followed by an attack. This also makes an estimate about packets involved in connection and the type of activity like less than five packets in a connection is defined as port scan. Between five and twelve packets in a connection is defined as vulnerability scan and more than twelve packets in a connections is defined as an attack. It also try to verify the reverse order for certain number of attackers and it found that more than 50% attacks were not preceded by a scan [66]. In experiment to know the behavior of attacker after compromising SSH, researchers found that most of attackers tried automated attack tools so having low level attacking skills and may be more interested in making money to sell these compromised machines to others. Only 22.09% attackers run any command after successful attack via SSH [65]. Downloading malicious code, running and installing rouge software, checking configuration details and changing password were few most common actions after a successful attack [65].

CHAPTER 3

RESEARCH METHODOLOGY

ProtoGENI is a representative of GENI functionality which provides an opportunity to explore the practical difficulties in achieving what researchers have planned and expect to achieve through GENI. Security is one of the most important issues and security services consist of Availability, Integrity, Authenticity, and authority to manage all security issues. Network or information security is important to minimize unauthorized access, misuse or denial to network functionality or network related accessible resources.

There are several issues which are of great importance to evaluate the functionality of ProtoGENI and GENI. ProtoGENI is a facility to test network research ideas or concepts to build a secure and accountable future Internet. Some basic questions arise by default: do we have infrastructure capable to conduct these research experiments? Do we have enough resources to support an expected set of concurrent research experiments? Do we have a manageable system for requesting, utilizing and releasing the network resources? Are there threats to disturb the normal resource management and availability of resources? Is traffic can be affected between slices and slivers? What can disturb a running network experiment? What are the limitations of using resources? What is the process to utilize the available resources at max? Is system working as it is supposed to work? Are there any unusual observations during experiments? Are there threats to modify or affect any running experiment by inside/ outside attacker? What would be the action in case of any component is compromised?

Resource management involves several security issues. First, who are authorized to use the resources? Which involves the question of identification and authorization; second, what are the availability or usage conflicts? ; third, is there interference between experiments?

How efficient is virtualization? Shared resources? Is there proper isolation between slices and slivers? Managing and securing virtualization to support the virtual networks and machines, and managing and securing the slices, is a question of resource management, and one critical to the success of GENI.

Recording network behavior can be a major help in controlling and regulating network traffic but to measure and record everything, including background traffic and timings, leads to privacy issues. The multi-national federation of networks forming GENI exacerbates this conflict.

On the basis of questions and issues about ProtoGENI, this research work focuses on experiments to observe and analyze ProtoGENI functionality and to identifying the security issues in ProtoGENI.

This research work tried to get answers for following three main research questions and their associated sub-problems:

RQ1: Is there any threat to ProtoGENI resources?

1. Outage of resources
2. Non-usability of available resources
3. Vulnerability of wildcard specifications in RSpecs

RQ2: Is there any problem in run-time interactions between ProtoGENI components?

1. Communication between slices/ slivers
2. Interactions between component managers
3. Traffic load and malicious traffic

RQ3: Is there any threat because of default settings for SSH and operating system vulnerabilities?

1. How default setting can threat the security in ProtoGENI?
2. How port scanning and vulnerability scanning can affect security of ProtoGENI system?
3. What can be done to improve the overall security of ProtoGENI?

In order to investigate above mentioned issues, it is necessary to know the facilities and functionality available with ProtoGENI. Availability and accessibility of ProtoGENI resources are of much significance as any experiment can be executed only if resources are available as per experiment’s requirement, for the required time and with consistent performance. Any loose end will put a doubt on the output of experiment and any small interference because of traffic load or malicious traffic can disturb other experiments. This work focuses on capturing different issues which can affect the basic functionality of ProtoGENI, and which can affect the whole GENI system eventually. In next chapter, different experiments will be designed and executed to gather information. Experiments try to explore the possible threats or operational glitches. The documentation will help to repeat the experiments and a base for expanding experiments to evaluate more, and to improve the overall ProtoGENI functionality and security.

CHAPTER 4

PROTOGENI MULTIPLE EXPERIMENTS

ProtoGENI is a setup to provide facilities to researchers to experiment with innovative network research. Before going ahead with different network research experiments, it is important to understand the infrastructure and mechanism to use ProtoGENI resources. These experiments evaluate the available options, test scripts and basic RSpec to initiate basic experiments to observe the ProtoGENI functionality, and also to observe the problems during different ProtoGENI functions. First step is to go through basic setup and functionality; second experiments will explore the different security questions related to ProtoGENI working on research questions.

Security architecture for any system includes a number of security services like confidentiality, Integrity, availability, etc. [5] Roles, responsibilities, authorization, and authentication mechanisms in the system are also crucial for robust security. ProtoGENI experiments are divided into four basic categories:

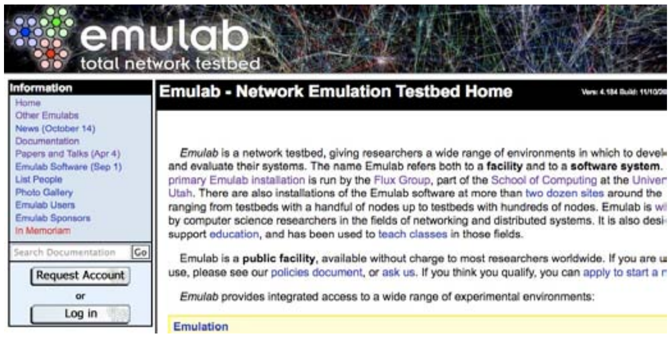
1. ProtoGENI setup and mechanism for experiments
    a. Initial setup
    b. Test scripts
    c. RSpecs
2. Threats to availability of ProtoGENI resources
3. Problems in run-time interactions between ProtoGENI components
4. Security threats at host machine in ProtoGENI setup

ProtoGENI Setup and Mechanism for Experiments

ProtoGENI can be used only by registered users on Emulab. There are many steps to complete before reaching to design or execute an experiment for network behavior analysis. Getting an Emulab account is the very first step to start the experimenting process.

Emulab Account and Project Membership

A new academic Emulab [9] project is generally initiated by project head which normally your advisor in research works or your professor in your course work which acts as your project leader. Project leader can associate others to be a part of project team. As shown in Fig. 2, visit https://www.emulab.net/ [9] to apply for an Emulab account. All project members have to send the request to be associated on a particular project and project leader approves them to be or not to be the part of project team.



Emulab is a network testbed, giving researchers a wide range of environments in which to devel
and evaluate their systems. The name Emulab refers both to a **facility** and to a **software system**.
primary Emulab installation is run by the Flux Group, part of the School of Computing at the Univer
Utah. There are also installations of the Emulab software at more than two dozen sites around the
ranging from testbeds with a handful of nodes up to testbeds with hundreds of nodes. Emulab is wi
by computer science researchers in the fields of networking and distributed systems. It is also desi
support education, and has been used to teach classes in those fields.

Emulab is a **public facility**, available without charge to most researchers worldwide. If you are u
use, please see our policies document, or ask us. If you think you qualify, you can apply to start a r

Emulab provides integrated access to a wide range of experimental environments:

Fig. 4.1: Emulab Account creation

Though there is no requirement to create a ProtoGENI experiment associated with your Emulab projects, initial account setup is done with association of some project initiated by your professor as project leader.

On host machine, a Windows 7 machine is used, and a virtual machine was created through VM player 3.0 and Ubuntu 9 was installed. As we will edit the files, and also need port scanner to find details about open ports and other things, we need to install related packages:

To edit files through command line

Sudo apt-get install vim

To get port scanner

Sudo apt-get install nmap

Sudo apt-get install zenmap

By default Ubuntu installs only OpenSSH –client software on host so server configuration file could not be located on host. OpenSSH-server was installed as

Sudo apt-get install openssh-server

To set stage for ProtoGENI slice and sliver creation, a short summary of required operations is as following in sequence to make it work right:

1) Install M2Crypto simply by running following command:

   sudo apt-get install m2crypto

2) Create directory *protogeni* and *test* as sub-directory of *protogeni*. Download test scripts from ProtoGENI website [55]to /protogeni/test by following command:

Home/protogeni/test *sudo wget* http://www.emulab.net/downloads/protogeni-tests.tar.gz

3) Now create .ssh and .ssl directories into home folder

4) Create protogeni-key pair by typing

   ssh-keygen -f protogeni-key

At the time of creating this key, system will ask for a passphrase, take time to choose a strong passphrase which is not easily guessable by attackers.

Go to your Emulab account- My Emulab- Profile- on left of this page user can see edit ssh keys, click on that [9]. Here user should upload the latest generated public key which can be browsed though available option. File to be uploaded here should be in home folder as protogeni-key.pub

On same page of Emulab account user can see ‘generate SSL certificate’ click on it and generate a SSL certificate. Now download it your host machine and save as encrypted.pem in .ssl directory. In .ssl directory, create a new file to save your passphrase in a file named as “password”. ProtoGENI test scripts will be able to read the passphrase from here.

5) User should run the python script to get the credential to be authorized for utilizing ProtoGENI resources.

   sudo python getcredential.py

   User can experience “WrongHost” problem as following statement:

   M2Crypto.SSL.Checker.WrongHost: Peer certificate commonName does not match host, expected boss.emulab.net, got www.emulab.net

   ProtoGENI solution: user can create a file $HOME/.protogeni-config.py with the following contents:

   XMLRPC_SERVER = {"default" : "www.emulab.net" } [28]

   Alternate Solution: User can make the same changes in *test-common.py* and it works fine.

6) User can verify the setup requirement by running python test scripts as following:

   Home/protogeni/test/ sudo python lookupuser.py

   Home/protogeni/test sudo python discover.py

7) If everything goes fine till here, user is all set to create ProtoGENI slices and slivers to do experiments.

8) Now the only thing to create a rspec.xml file as per requirement of experiment. Sliver will be created on the basis of resources defined in rspec file.

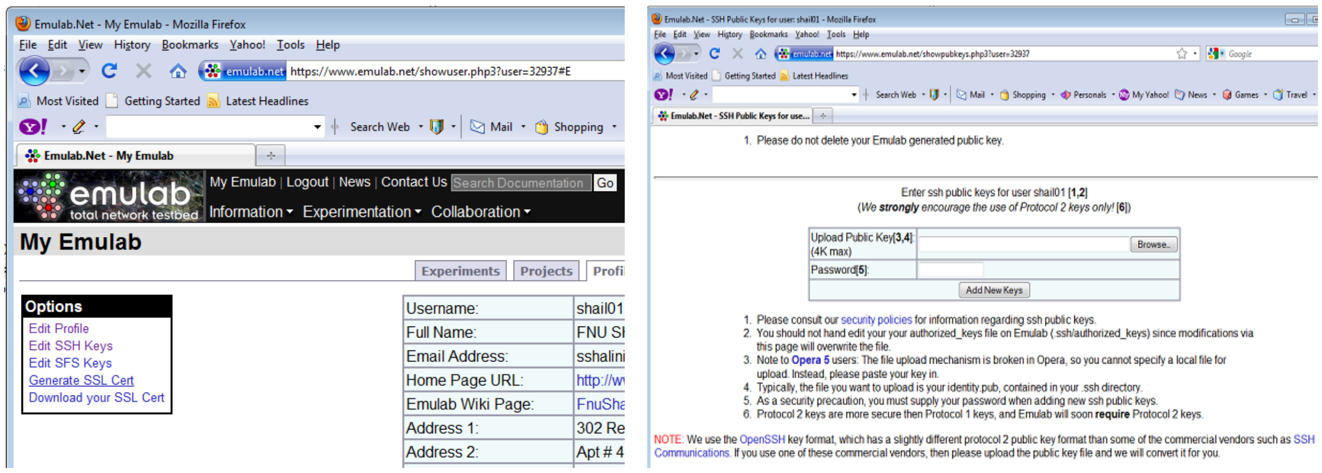
After initial setup, user can create slice/sliver. Any suitable name can be chosen for your slice, and it can be replaced at place of “myslice” in following command.:

python registerslice.py –n myslice

Now a sliver can be created with same slice name and RSpec file name as following:

python createsliver.py –n myslice shailrspec.xml

It will create a sliver with required resources defined in shailrspec.xml file, and it will return the details of machines provided to user with node identification, which represents Emulab remote machines.


Emulab.Net - My Emulab - Mozilla Firefox
File Edit View History Bookmarks Yahoo! Tools Help
https://www.emulab.net/showuser.php3?user=32937#E
Most Visited Getting Started Latest Headlines
Search Web Mail Shopping
Emulab.Net - My Emulab
emulab total network testbed
My Emulab | Logout | News | Contact Us Search Documentation Go
Information Experimentation Collaboration
Experiments Projects Profi
Options
Edit Profile
Edit SSH Keys
Edit SFS Keys
Generate SSL Cert
Download your SSL Cert
Emulab.Net - SSH Public Keys for user: shail01 - Mozilla Firefox
https://www.emulab.net/showpubkeys.php3?user=32937
Search Web Mail Shopping Personals My Yahoo! News Games Travel
Emulab.Net - SSH Public Keys for use...
Browse..
Add New Keys

My Emulab

Username:	shail01
Full Name:	FNU Sh
Email Address:	sshalini
Home Page URL:	http://w
Emulab Wiki Page:	FnuSha
Address 1:	302 Re
Address 2:	Apt # 4

1. Please do not delete your Emulab generated public key.

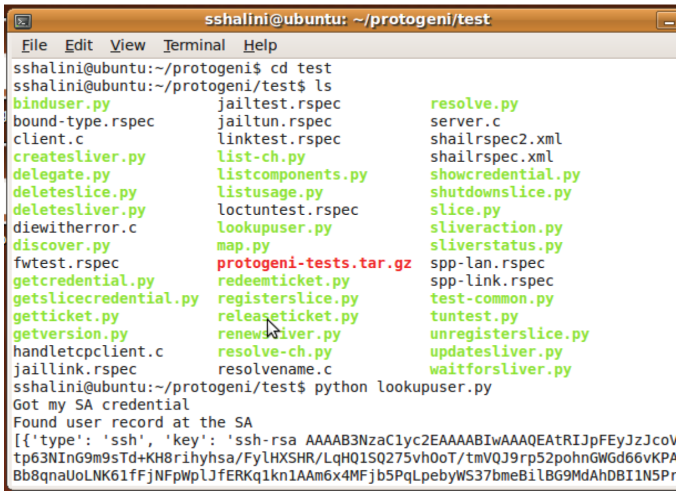
Enter ssh public keys for user shail01 [1,2]
(*We **strongly** encourage the use of Protocol 2 keys only!* [6])

Upload Public Key[3,4]: (4K max)	
Password[5]:	

1. Please consult our security policies for information regarding ssh public keys.
2. You should not hand edit your your authorized_keys file on Emulab (.ssh/authorized_keys) since modifications via this page will overwrite the file.
3. Note to **Opera 5** users: The file upload mechanism is broken in Opera, so you cannot specify a local file for upload. Instead, please paste your key in.
4. Typically, the file you want to upload is your identity.pub, contained in your .ssh directory.
5. As a security precaution, you must supply your password when adding new ssh public keys.
6. Protocol 2 keys are more secure then Protocol 1 keys, and Emulab will soon **require** Protocol 2 keys.

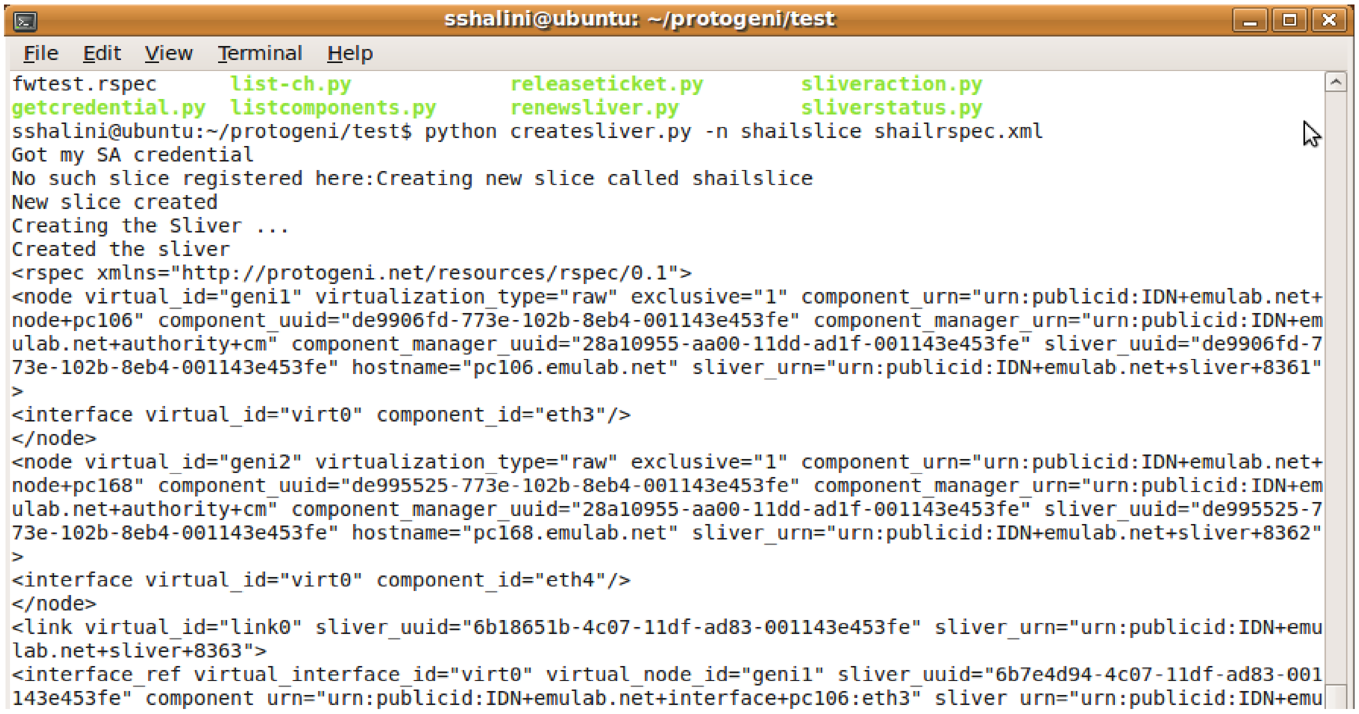
NOTE: We use the OpenSSH key format, which has a slightly different protocol 2 public key format than some of the commercial vendors such as SSH Communications. If you use one of these commercial vendors, then please upload the public key file and we will convert it for you.

Fig. 4.2: Generating a SSL certificate and uploading of SSH ProtoGENI keys to Emulab account



```
sshalini@ubuntu:~/protogeni$ cd test
sshalini@ubuntu:~/protogeni/test$ ls
binduser.py             jailtest.rspec          resolve.py
bound-type.rspec        jailtun.rspec           server.c
client.c                linktest.rspec          shailrspec2.xml
createsliver.py         list-ch.py              shailrspec.xml
delegate.py             listcomponents.py       showcredential.py
deleteslice.py          listusage.py            shutdownslice.py
deletesliver.py         loctuntest.rspec        slice.py
diewitherror.c          lookupuser.py           sliveraction.py
discover.py             map.py                  sliverstatus.py
fwtest.rspec            protogeni-tests.tar.gz  spp-lan.rspec
getcredential.py        redeemticket.py         spp-link.rspec
getslicecredential.py   registerslice.py        test-common.py
getticket.py            releaseticket.py        tuntest.py
getversion.py           renewsliver.py          unregisterslice.py
handletcpclient.c       resolve-ch.py           updatesliver.py
jaillink.rspec          resolvename.c           waitforsliver.py
sshalini@ubuntu:~/protogeni/test$ python lookupuser.py
Got my SA credential
Found user record at the SA
[{'type': 'ssh', 'key': 'ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAtRIJpFEyJzJcoV
tp63NInG9m9sTd+KH8rihyhsa/FylHXSHR/LqHQ1SQ275vhOoT/tmVQJ9rp52pohnGWGd66vKPA
Bb8qnaUoLNK61fFjNFpWplJfERKq1kn1AAm6x4MFjb5PqLpebyWS37bmeBilBG9MdAhDBI1N5Pr
```

Fig. 4.3: Verification of initial settings for ProtoGENI



```
fwtest.rspec      list-ch.py            releaseticket.py        sliveraction.py
getcredential.py  listcomponents.py     renewsliver.py          sliverstatus.py
sshalini@ubuntu:~/protogeni/test$ python createsliver.py -n shailslice shailrspec.xml
Got my SA credential
No such slice registered here:Creating new slice called shailslice
New slice created
Creating the Sliver ...
Created the sliver
<rspec xmlns="http://protogeni.net/resources/rspec/0.1">
<node virtual_id="geni1" virtualization_type="raw" exclusive="1" component_urn="urn:publicid:IDN+emulab.net+
node+pc106" component_uuid="de9906fd-773e-102b-8eb4-001143e453fe" component_manager_urn="urn:publicid:IDN+em
ulab.net+authority+cm" component_manager_uuid="28a10955-aa00-11dd-ad1f-001143e453fe" sliver_uuid="de9906fd-7
73e-102b-8eb4-001143e453fe" hostname="pc106.emulab.net" sliver_urn="urn:publicid:IDN+emulab.net+sliver+8361"
>
<interface virtual_id="virt0" component_id="eth3"/>
</node>
<node virtual_id="geni2" virtualization_type="raw" exclusive="1" component_urn="urn:publicid:IDN+emulab.net+
node+pc168" component_uuid="de995525-773e-102b-8eb4-001143e453fe" component_manager_urn="urn:publicid:IDN+em
ulab.net+authority+cm" component_manager_uuid="28a10955-aa00-11dd-ad1f-001143e453fe" sliver_uuid="de995525-7
73e-102b-8eb4-001143e453fe" hostname="pc168.emulab.net" sliver_urn="urn:publicid:IDN+emulab.net+sliver+8362"
>
<interface virtual_id="virt0" component_id="eth4"/>
</node>
<link virtual_id="link0" sliver_uuid="6b18651b-4c07-11df-ad83-001143e453fe" sliver_urn="urn:publicid:IDN+emu
lab.net+sliver+8363">
<interface_ref virtual_interface_id="virt0" virtual_node_id="geni1" sliver_uuid="6b7e4d94-4c07-11df-ad83-001
143e453fe" component_urn="urn:publicid:IDN+emulab.net+interface+pc106:eth3" sliver_urn="urn:publicid:IDN+emu
```

Fig. 4.4: Creating the sliver with resource specifications

Now sliver is created and we have remote machines details, we can login to those machines by following command.

ssh shail01@pc168.emulab.net

pc168 should be replaced as per acquired pc-ID in RSpec file or as per allotted by the system.

ProtoGENI Resources Acquisition, Utilization, and Releasing through Test Scripts

The setup of ProtoGENI and designed test scripts are supposed to help in understanding the operation of ProtoGENI. These test scripts provide different operations to acquire and release network resources to execute network experiments.

This part of experiments is an effort to evaluate the current settings and usage of these test scripts. Simple experiments are done to see the applicability of different test scripts. Observations have been recorded about redundant, and also for not in operation test scripts. This work can help a novice learner to understand ProtoGENI functionality and associated problems a bit more with practical outputs and results. It can help to initiate, and to follow a straightforward path where user can see the application of each test script, redundancy of operations and rights available to use resources. This work also includes the observations about inconsistency in outputs and not getting the intended operations.

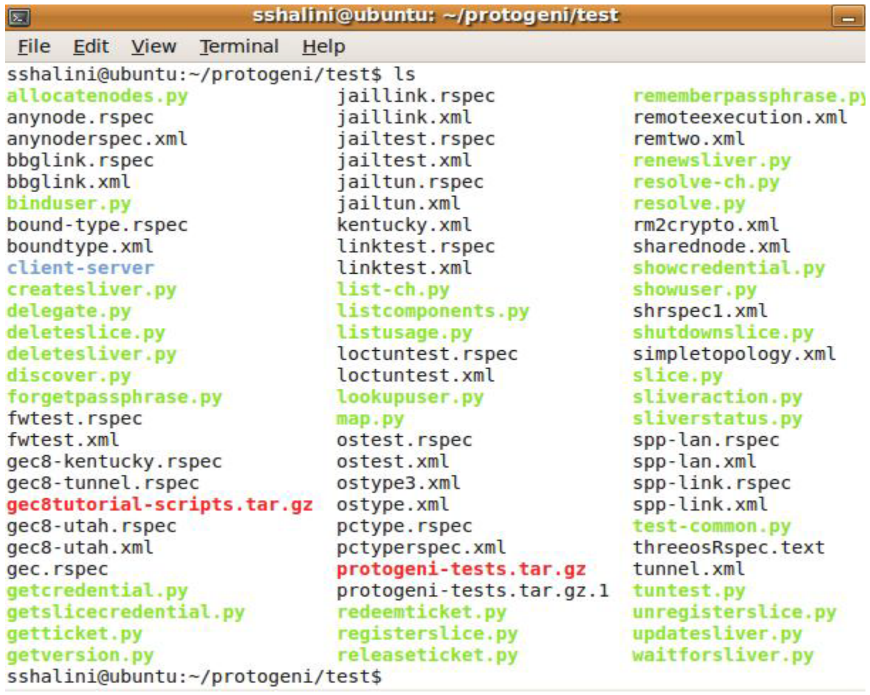
Emulab resources are being used for these operations. Utah Emulab is primary component manager and many other component mangers exist. There are different kinds of machines available which are helpful to do specific experiments to see the working on that particular type of machine and also useful to verify the compatibility issues among different type of machines. We tried to explore the usability and applicability of test scripts provided by ProtoGENI.

As we are concerned to identifying the loose ends, and to improve descriptions to follow steps with more ease, we tried to execute the available test scripts to see their effect on experiments and what information may be more helpful to be available through test scripts.

Sometimes a sequence of events of absence of some steps may not be a problem for scholars but it can significantly affect the learning curve of a novice.

Test Scripts in ProtoGENI

A suite of command-line tools are now available for operation with ProtoGENI services. They act as an interim user interface, to provide a means to control ProtoGENI facilities until more sophisticated tools are available; and they are also a convenient debugging mechanism to test the emerging ProtoGENI components.


sshalini@ubuntu: ~/protogeni/test
File Edit View Terminal Help

```
sshalini@ubuntu:~/protogeni/test$ ls
allocatenodes.py             jaillink.rspec             rememberpassphrase.py
anynode.rspec                jaillink.xml               remoteexecution.xml
anynoderspec.xml             jailtest.rspec             remtwo.xml
bbglink.rspec                jailtest.xml               renewsliver.py
bbglink.xml                  jailtun.rspec              resolve-ch.py
binduser.py                  jailtun.xml                resolve.py
bound-type.rspec             kentucky.xml               rm2crypto.xml
boundtype.xml                linktest.rspec             sharednode.xml
client-server                linktest.xml               showcredential.py
createsliver.py              list-ch.py                 showuser.py
delegate.py                  listcomponents.py          shrspec1.xml
deleteslice.py               listusage.py               shutdownslice.py
deletesliver.py              loctuntest.rspec           simpletopology.xml
discover.py                  loctuntest.xml             slice.py
forgetpassphrase.py          lookupuser.py              sliveraction.py
fwtest.rspec                 map.py                     sliverstatus.py
fwtest.xml                   ostest.rspec               spp-lan.rspec
gec8-kentucky.rspec          ostest.xml                 spp-lan.xml
gec8-tunnel.rspec            ostype3.xml                spp-link.rspec
gec8tutorial-scripts.tar.gz  ostype.xml                 spp-link.xml
gec8-utah.rspec              pctype.rspec               test-common.py
gec8-utah.xml                pctyperspec.xml            threeosRspec.text
gec.rspec                    protogeni-tests.tar.gz     tunnel.xml
getcredential.py             protogeni-tests.tar.gz.1   tuntest.py
getslicecredential.py        redeemticket.py            unregisterslice.py
getticket.py                 registerslice.py           updatesliver.py
getversion.py                releaseticket.py           waitforsliver.py
sshalini@ubuntu:~/protogeni/test$
```

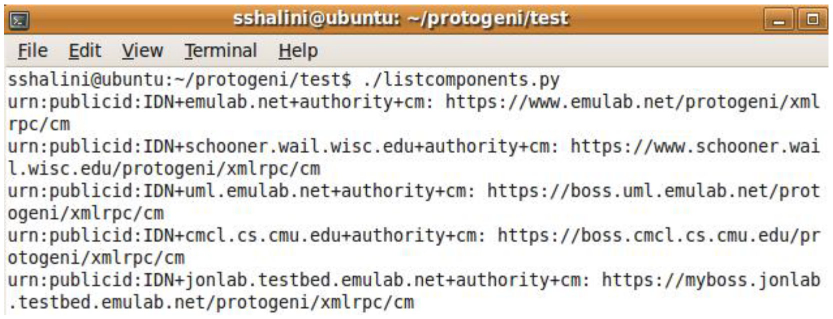
Fig. 4.5: Test scripts suite available in ProtoGENI

All python test scripts help in doing different ProtoGENI operations. Slice Authority handles some operations include unprivileged operations, information operations, registration operations, and few special operations. Component manager interacts with user, SA, and clearinghouse through some information operations, ticket operations, manipulation operations, and special operations.

A. Initialization verification scripts

1) *LookUpUser:* User can verify that things have worked properly by running the *lookupuser.py* test script and checking that the contents of the *protogeni*`-key.pub` file are included in the output: these are the same keys that *`createsliver.py`* and *`redeemticket.py`* will pass on to the nodes in the slivers it requests [28].

2) *ListComponents:* listcomponents.py provides details of active Component Managers including URN and URL.


sshalini@ubuntu: ~/protogeni/test
File Edit View Terminal Help

```
sshalini@ubuntu:~/protogeni/test$ ./listcomponents.py
urn:publicid:IDN+emulab.net+authority+cm: https://www.emulab.net/protogeni/xml
rpc/cm
urn:publicid:IDN+schooner.wail.wisc.edu+authority+cm: https://www.schooner.wai
l.wisc.edu/protogeni/xmlrpc/cm
urn:publicid:IDN+uml.emulab.net+authority+cm: https://boss.uml.emulab.net/prot
ogeni/xmlrpc/cm
urn:publicid:IDN+cmcl.cs.cmu.edu+authority+cm: https://boss.cmcl.cs.cmu.edu/pr
otogeni/xmlrpc/cm
urn:publicid:IDN+jonlab.testbed.emulab.net+authority+cm: https://myboss.jonlab
.testbed.emulab.net/protogeni/xmlrpc/cm
```

Fig. 4.6: List of active Component Managers

3) *test-common.py* script is being used by all other scripts if attacker has access to the physical machine he can inject simple messages as print outputs without disturbing the actual process of script to fool users. If it is not inserted in proper place, it can affect normal proceedings of the test script and will generate some error to warn user or he/she can look for solutions.

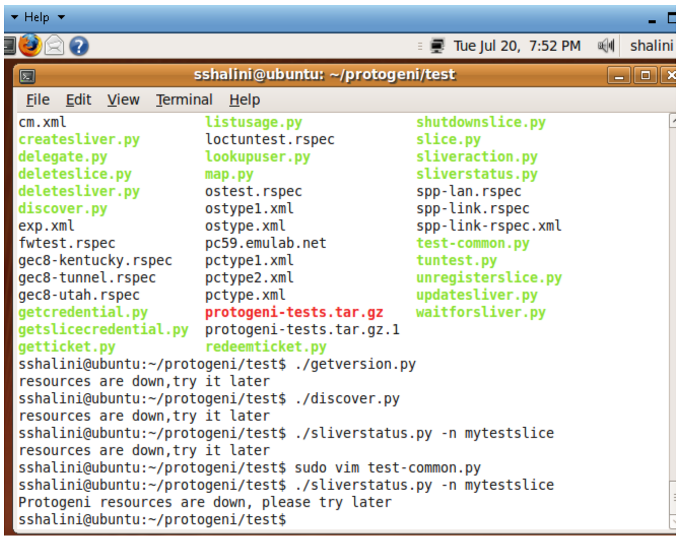
If user sees a simple printing message in a convincing language, it can halt any further effort by user to request any resource or to do any other operation. Print messages were inserted at different places but it was showing errors and messages about checking those particular lines of code, simply ignoring the message, and showing the message but doing normal process too. After couple of efforts, we modified the test-common.py and got the convincible output when we tried to execute any other test script.

Following code lines were inserted in the end of the file

print"ProtoGENI resources are down, please try later"

exit()

As per fig. 7, user may assume that something in process so resources are not available, and it might restrict any further effort from user to investigate the actual problem.


Help
Tue Jul 20, 7:52 PM
shalini
sshalini@ubuntu: ~/protogeni/test
File Edit View Terminal Help

```
cm.xml                 listusage.py           shutdownslice.py
createsliver.py        loctuntest.rspec       slice.py
delegate.py            lookupuser.py          sliveraction.py
deleteslice.py         map.py                 sliverstatus.py
deletesliver.py        ostest.rspec           spp-lan.rspec
discover.py            ostype1.xml            spp-link.rspec
exp.xml                ostype.xml             spp-link-rspec.xml
fwtest.rspec           pc59.emulab.net        test-common.py
gec8-kentucky.rspec    pctype1.xml            tuntest.py
gec8-tunnel.rspec      pctype2.xml            unregisterslice.py
gec8-utah.rspec        pctype.xml             updatesliver.py
getcredential.py       protogeni-tests.tar.gz waitforsliver.py
getslicecredential.py  protogeni-tests.tar.gz.1
getticket.py           redeemticket.py
sshalini@ubuntu:~/protogeni/test$ ./getversion.py
resources are down,try it later
sshalini@ubuntu:~/protogeni/test$ ./discover.py
resources are down,try it later
sshalini@ubuntu:~/protogeni/test$ ./sliverstatus.py -n mytestslice
resources are down,try it later
sshalini@ubuntu:~/protogeni/test$ sudo vim test-common.py
sshalini@ubuntu:~/protogeni/test$ ./sliverstatus.py -n mytestslice
Protogeni resources are down, please try later
sshalini@ubuntu:~/protogeni/test$
```

Fig. 4.7: System is unable to run any script after changes in test-common.py

B. Slice Authority Operations

1) Unprivileged operations: *A valid ProtoGENI certificate is required to establish the SSL connection.*

GetCredential: ProtoGENICredentials are signed documents which describe privileges held by a principal (a user). Conceptually, they identify the owning user and a target ProtoGENI object to which the privileges apply.

Since you need a credential to do most anything in ProtoGENI, you first have to get your "self" credential, which is just a generic credential that gives you permission to do basic kinds of things, like register a slice name.

Your self-credential is issued by your local Emulab. For example, to get your self-credential, you would run the getcredential.py test script on `usernames [28].`

2) Information Operations

DiscoverResources: this operation gives the information of all available resources at Component Manager.

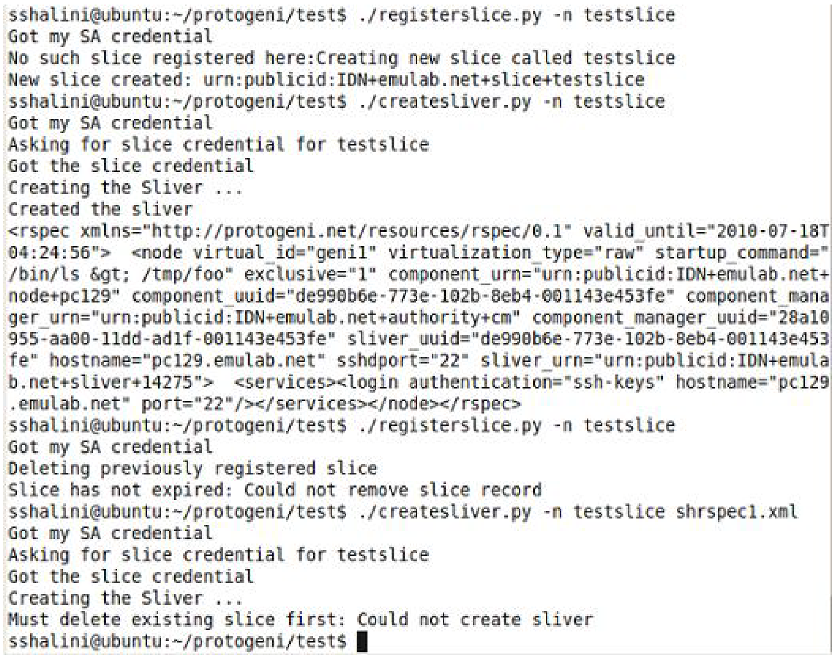
./discover.py

3) Registration operations

Registerslice.py registers a slice name with slice authority but do not provide any resources to use. Createsliver.py creates a sliver and gets single node resource from component manager if no RSpec .xml file is specified in the command.

If a researcher tries to register another slice with same name, system checks the status of existing slice and returns the status accordingly as shown in figure 8.

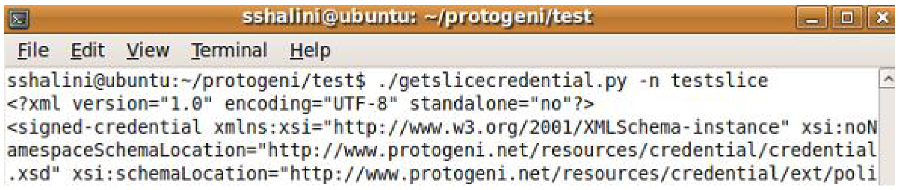
Required resources for an experiment can be defined in a RSpec .xml file and a sliver can be created with that specific .xml file but only one sliver can be active at a time so user should delete the existing sliver to create a new one which we will describe in component manager manipulation operations.



```
sshalini@ubuntu:~/protogeni/test$ ./registerslice.py -n testslice
Got my SA credential
No such slice registered here:Creating new slice called testslice
New slice created: urn:publicid:IDN+emulab.net+slice+testslice
sshalini@ubuntu:~/protogeni/test$ ./createsliver.py -n testslice
Got my SA credential
Asking for slice credential for testslice
Got the slice credential
Creating the Sliver ...
Created the sliver
<rspec xmlns="http://protogeni.net/resources/rspec/0.1" valid_until="2010-07-18T
04:24:56">  <node virtual_id="geni1" virtualization_type="raw" startup_command="
/bin/ls &gt; /tmp/foo" exclusive="1" component_urn="urn:publicid:IDN+emulab.net+
node+pc129" component_uuid="de990b6e-773e-102b-8eb4-001143e453fe" component_mana
ger_urn="urn:publicid:IDN+emulab.net+authority+cm" component_manager_uuid="28a10
955-aa00-11dd-ad1f-001143e453fe" sliver_uuid="de990b6e-773e-102b-8eb4-001143e453
fe" hostname="pc129.emulab.net" sshdport="22" sliver_urn="urn:publicid:IDN+emula
b.net+sliver+14275">  <services><login authentication="ssh-keys" hostname="pc129
.emulab.net" port="22"/></services></node></rspec>
sshalini@ubuntu:~/protogeni/test$ ./registerslice.py -n testslice
Got my SA credential
Deleting previously registered slice
Slice has not expired: Could not remove slice record
sshalini@ubuntu:~/protogeni/test$ ./createsliver.py -n testslice shrspec1.xml
Got my SA credential
Asking for slice credential for testslice
Got the slice credential
Creating the Sliver ...
Must delete existing slice first: Could not create sliver
sshalini@ubuntu:~/protogeni/test$ 
```

Fig. 4.8: Slice registration and creation of sliver

GetSliceCredential : getslicecredential.py generates slice details of specified slice including expiring time with other details.



sshalini@ubuntu: ~/protogeni/test
File Edit View Terminal Help

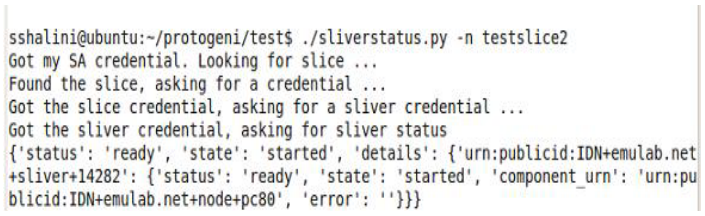
```
sshalini@ubuntu:~/protogeni/test$ ./getslicecredential.py -n testslice
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<signed-credential xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noN
amespaceSchemaLocation="http://www.protogeni.net/resources/credential/credential
.xsd" xsi:schemaLocation="http://www.protogeni.net/resources/credential/ext/poli
```

Fig. 4.9: To get the slice credentials

C. Component Manager Operations

1) Information Operations

SliverStatus: sliverstatus.py script tells about the current status of sliver resources. It has details about sliver resources and their status like whether a node is ready or in changing status mode.

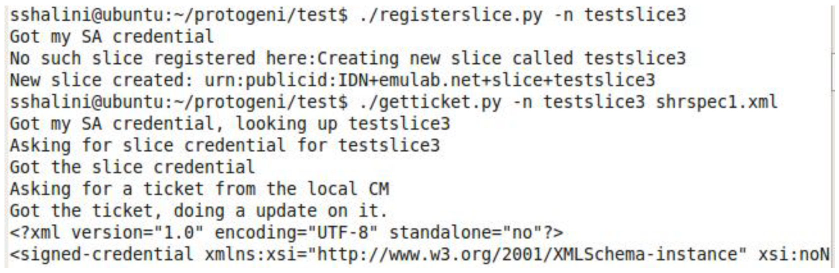


```
sshalini@ubuntu:~/protogeni/test$ ./sliverstatus.py -n testslice2
Got my SA credential. Looking for slice ...
Found the slice, asking for a credential ...
Got the slice credential, asking for a sliver credential ...
Got the sliver credential, asking for sliver status
{'status': 'ready', 'state': 'started', 'details': {'urn:publicid:IDN+emulab.net
+sliver+14282': {'status': 'ready', 'state': 'started', 'component_urn': 'urn:pu
blicid:IDN+emulab.net+node+pc80', 'error': ''}}}
```

Fig. 4.10: sliverstatus.py to know the status of a sliver

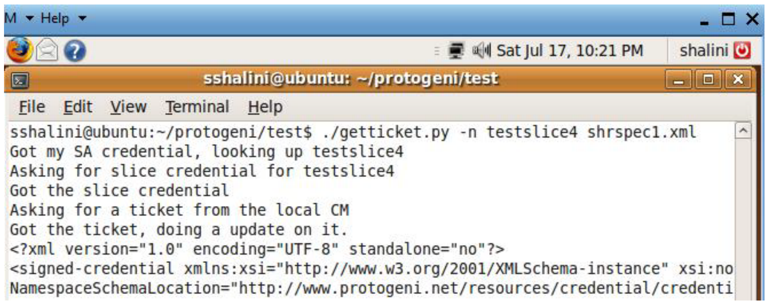
2) Ticket Operations

GetTicket: getticket.py script provides a promise to assign you the defined resources in your associated RSpec .xml file in command but user can hold control on those resources only after redemption of ticket before expiring of the time limit defined in details in output after running this script but practically it is not working as per this time. Ticket has to be redeemed few minutes after getting the ticket otherwise it says do not have any ticket as shown in fig with time details captured on screen. In our trial, no ticket was found for slice after 5 minutes.



```
sshalini@ubuntu:~/protogeni/test$ ./registerslice.py -n testslice3
Got my SA credential
No such slice registered here:Creating new slice called testslice3
New slice created: urn:publicid:IDN+emulab.net+slice+testslice3
sshalini@ubuntu:~/protogeni/test$ ./getticket.py -n testslice3 shrspec1.xml
Got my SA credential, looking up testslice3
Asking for slice credential for testslice3
Got the slice credential
Asking for a ticket from the local CM
Got the ticket, doing a update on it.
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<signed-credential xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noN
```

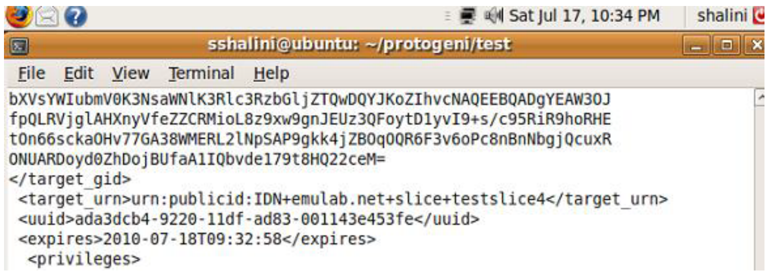
Fig. 4.11: getticket.py after registering a slice


M Help
Sat Jul 17, 10:21 PM
shalini
sshalini@ubuntu: ~/protogeni/test
File Edit View Terminal Help

```
sshalini@ubuntu:~/protogeni/test$ ./getticket.py -n testslice4 shrspec1.xml
Got my SA credential, looking up testslice4
Asking for slice credential for testslice4
Got the slice credential
Asking for a ticket from the local CM
Got the ticket, doing a update on it.
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<signed-credential xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:no
NamespaceSchemaLocation="http://www.protogeni.net/resources/credential/credenti
```

Fig. 4.12: getticket.py without registering a slice first

Testslice4 was tried to redeem ticket quickly just after getting the ticket and redemption of ticket was tried after 5 minutes for testslice3. We can see in fig. 13 that expiring time of sliver testslice4 is apprx. 11 hours while ticket could not be redeemed for testslice3 after 5 minutes.

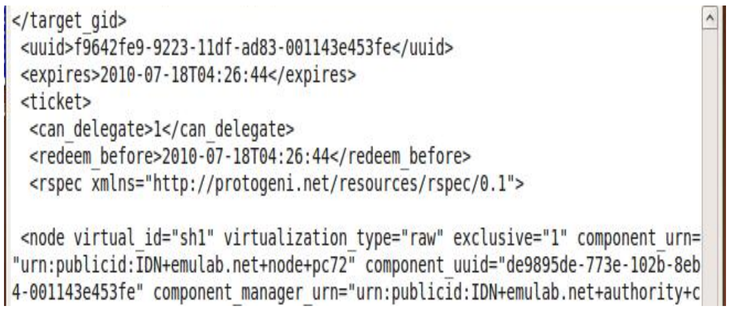

Sat Jul 17, 10:34 PM
shalini
sshalini@ubuntu: ~/protogeni/test
File Edit View Terminal Help

```
bXVsYWIubmV0K3NsaWNlK3Rlc3RzbGljZTQwDQYJKoZIhvcNAQEEBQADgYEAW3OJ
fpQLRVjglAHXnyVfeZZCRMioL8z9xw9gnJEUz3QFoytD1yvI9+s/c95RiR9hoRHE
tOn66sckaOHv77GA38WMERL2lNpSAP9gkk4jZBOq0QR6F3v6oPc8nBnNbgjQcuxR
ONUARDoyd0ZhDojBUfaA1IQbvde179t8HQ22ceM=
</target_gid>
 <target_urn>urn:publicid:IDN+emulab.net+slice+testslice4</target_urn>
 <uuid>ada3dcb4-9220-11df-ad83-001143e453fe</uuid>
 <expires>2010-07-18T09:32:58</expires>
  <privileges>
```

Fig. 4.13: getslicecredential.py details of testslice4 after redemption of ticket

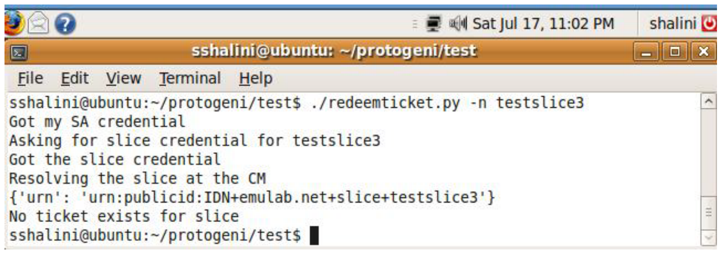
Redeeming a ticket creates a sliver, and returns another credential to you. The sliver is the portion of the computing resource that has been granted to the user [28].

Though Redemption of ticket provide a similar active sliver which can be created through createsliver.py, it is a bit lengthy and quick series of operations to keep you sliver active for a required amount of time for executing experiments.



```
</target_gid>
 <uuid>f9642fe9-9223-11df-ad83-001143e453fe</uuid>
 <expires>2010-07-18T04:26:44</expires>
 <ticket>
  <can_delegate>1</can_delegate>
  <redeem_before>2010-07-18T04:26:44</redeem_before>
  <rspec xmlns="http://protogeni.net/resources/rspec/0.1">

 <node virtual_id="sh1" virtualization_type="raw" exclusive="1" component_urn=
"urn:publicid:IDN+emulab.net+node+pc72" component_uuid="de9895de-773e-102b-8eb
4-001143e453fe" component_manager_urn="urn:publicid:IDN+emulab.net+authority+c
```

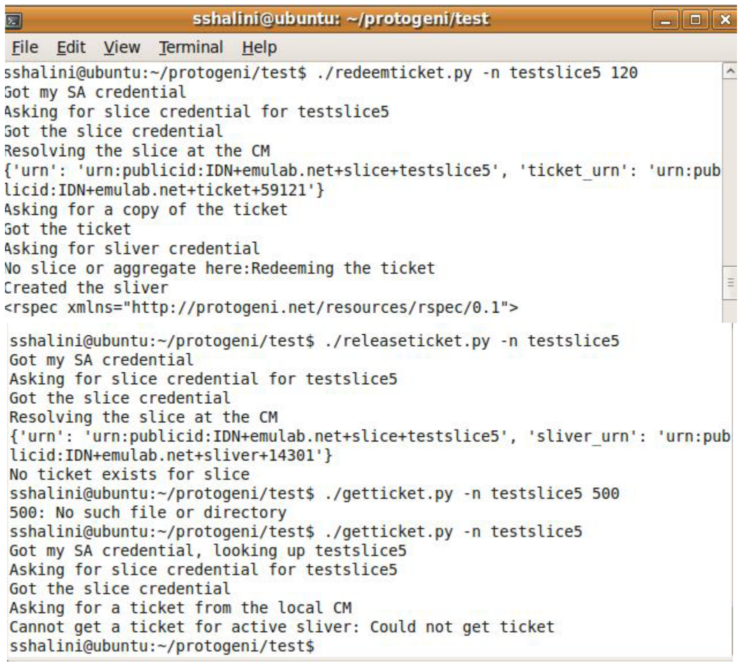
Fig. 4.14: Details for getticket and supposed time to redeem the ticket for testslice3



```
sshalini@ubuntu:~/protogeni/test$ ./redeemticket.py -n testslice3
Got my SA credential
Asking for slice credential for testslice3
Got the slice credential
Resolving the slice at the CM
{'urn': 'urn:publicid:IDN+emulab.net+slice+testslice3'}
No ticket exists for slice
sshalini@ubuntu:~/protogeni/test$
```

Fig. 4.15: No ticket exists after five minutes of getting ticket for testslice3

We executed getticket.py, redeemticket.py, releaseticket.py and getticket.py one after another to see the sequence of processes. After redemption of a ticket the slice got an active sliver and then no ticket operation works on active sliver.



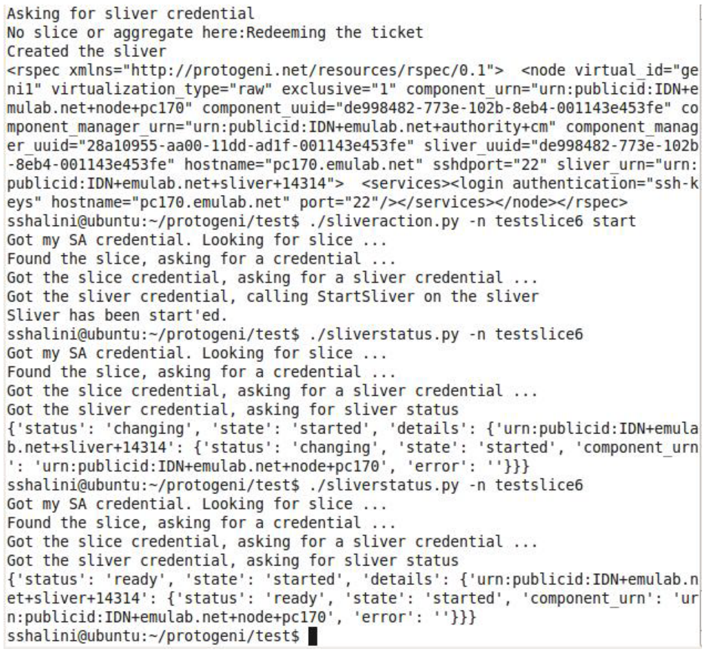
```
sshalini@ubuntu:~/protogeni/test$ ./redeemticket.py -n testslice5 120
Got my SA credential
Asking for slice credential for testslice5
Got the slice credential
Resolving the slice at the CM
{'urn': 'urn:publicid:IDN+emulab.net+slice+testslice5', 'ticket_urn': 'urn:pub
licid:IDN+emulab.net+ticket+59121'}
Asking for a copy of the ticket
Got the ticket
Asking for sliver credential
No slice or aggregate here:Redeeming the ticket
Created the sliver
<rspec xmlns="http://protogeni.net/resources/rspec/0.1">
```

```
sshalini@ubuntu:~/protogeni/test$ ./releaseticket.py -n testslice5
Got my SA credential
Asking for slice credential for testslice5
Got the slice credential
Resolving the slice at the CM
{'urn': 'urn:publicid:IDN+emulab.net+slice+testslice5', 'sliver_urn': 'urn:pub
licid:IDN+emulab.net+sliver+14301'}
No ticket exists for slice
sshalini@ubuntu:~/protogeni/test$ ./getticket.py -n testslice5 500
500: No such file or directory
sshalini@ubuntu:~/protogeni/test$ ./getticket.py -n testslice5
Got my SA credential, looking up testslice5
Asking for slice credential for testslice5
Got the slice credential
Asking for a ticket from the local CM
Cannot get a ticket for active sliver: Could not get ticket
sshalini@ubuntu:~/protogeni/test$
```

Fig. 4.16: No ticket operation after redemption of ticket as sliver act as active

3) Manipulation Operations

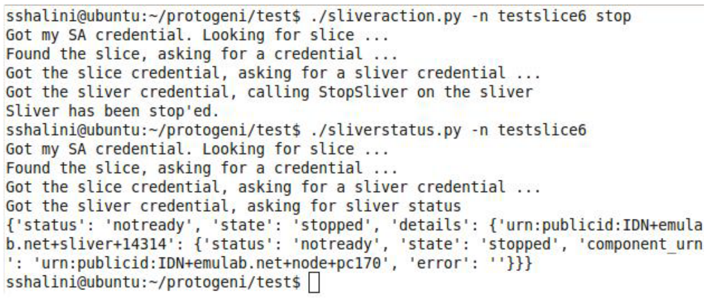
SliverAction: *sliveraction.py* helps in starting, restarting or to stop a sliver. After redeeming your tickets, you need to *start* your slivers. This is because while the computing resources have been bound to your slice, but they have not been initialized. [28][38].



```
Asking for sliver credential
No slice or aggregate here:Redeeming the ticket
Created the sliver
<rspec xmlns="http://protogeni.net/resources/rspec/0.1">  <node virtual_id="ge
ni1" virtualization_type="raw" exclusive="1" component_urn="urn:publicid:IDN+e
mulab.net+node+pc170" component_uuid="de998482-773e-102b-8eb4-001143e453fe" co
mponent_manager_urn="urn:publicid:IDN+emulab.net+authority+cm" component_manag
er_uuid="28a10955-aa00-11dd-ad1f-001143e453fe" sliver_uuid="de998482-773e-102b
-8eb4-001143e453fe" hostname="pc170.emulab.net" sshdport="22" sliver_urn="urn:
publicid:IDN+emulab.net+sliver+14314">  <services><login authentication="ssh-k
eys" hostname="pc170.emulab.net" port="22"/></services></node></rspec>
sshalini@ubuntu:~/protogeni/test$ ./sliveraction.py -n testslice6 start
Got my SA credential. Looking for slice ...
Found the slice, asking for a credential ...
Got the slice credential, asking for a sliver credential ...
Got the sliver credential, calling StartSliver on the sliver
Sliver has been start'ed.
sshalini@ubuntu:~/protogeni/test$ ./sliverstatus.py -n testslice6
Got my SA credential. Looking for slice ...
Found the slice, asking for a credential ...
Got the slice credential, asking for a sliver credential ...
Got the sliver credential, asking for sliver status
{'status': 'changing', 'state': 'started', 'details': {'urn:publicid:IDN+emula
b.net+sliver+14314': {'status': 'changing', 'state': 'started', 'component_urn
': 'urn:publicid:IDN+emulab.net+node+pc170', 'error': ''}}}
sshalini@ubuntu:~/protogeni/test$ ./sliverstatus.py -n testslice6
Got my SA credential. Looking for slice ...
Found the slice, asking for a credential ...
Got the slice credential, asking for a sliver credential ...
Got the sliver credential, asking for sliver status
{'status': 'ready', 'state': 'started', 'details': {'urn:publicid:IDN+emulab.n
et+sliver+14314': {'status': 'ready', 'state': 'started', 'component_urn': 'ur
n:publicid:IDN+emulab.net+node+pc170', 'error': ''}}}
sshalini@ubuntu:~/protogeni/test$ █
```

Fig. 4.17: Starting of sliver after redemption of ticket and verification of sliver status

User can also stop the sliver with sliveraction.py command as shown in fig 4.18.

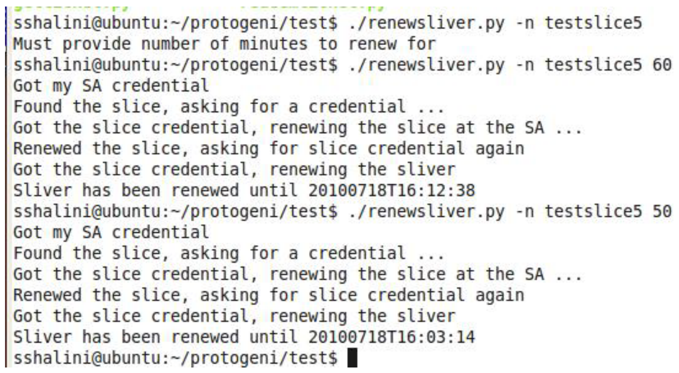


```
sshalini@ubuntu:~/protogeni/test$ ./sliveraction.py -n testslice6 stop
Got my SA credential. Looking for slice ...
Found the slice, asking for a credential ...
Got the slice credential, asking for a sliver credential ...
Got the sliver credential, calling StopSliver on the sliver
Sliver has been stop'ed.
sshalini@ubuntu:~/protogeni/test$ ./sliverstatus.py -n testslice6
Got my SA credential. Looking for slice ...
Found the slice, asking for a credential ...
Got the slice credential, asking for a sliver credential ...
Got the sliver credential, asking for sliver status
{'status': 'notready', 'state': 'stopped', 'details': {'urn:publicid:IDN+emula
b.net+sliver+14314': {'status': 'notready', 'state': 'stopped', 'component_urn
': 'urn:publicid:IDN+emulab.net+node+pc170', 'error': ''}}}
sshalini@ubuntu:~/protogeni/test$ 
```

Fig. 4.18: stopping a sliver through sliveraction.py

If user have any problem with sliver resources and want to reboot the nodes, user can place restart in the command in place of start/ stop.

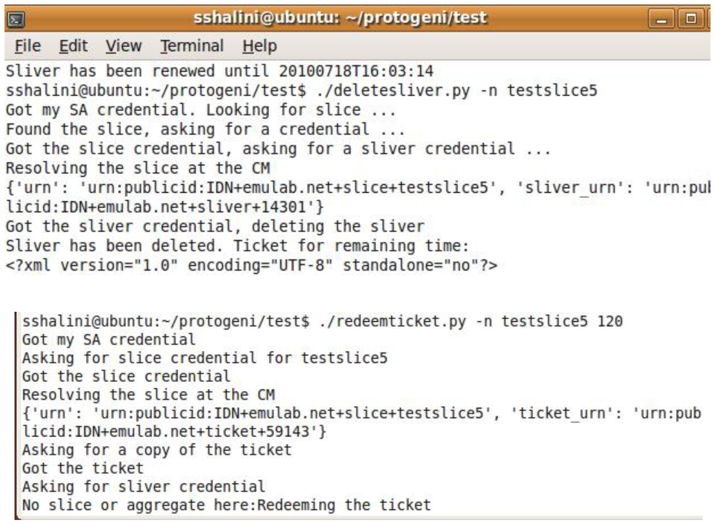
RenewSliver: renewsliver.py helps in extending the expiring time of a sliver and availability of associated resources for a specified time (in minutes) given in the command.



```
sshalini@ubuntu:~/protogeni/test$ ./renewsliver.py -n testslice5
Must provide number of minutes to renew for
sshalini@ubuntu:~/protogeni/test$ ./renewsliver.py -n testslice5 60
Got my SA credential
Found the slice, asking for a credential ...
Got the slice credential, renewing the slice at the SA ...
Renewed the slice, asking for slice credential again
Got the slice credential, renewing the sliver
Sliver has been renewed until 20100718T16:12:38
sshalini@ubuntu:~/protogeni/test$ ./renewsliver.py -n testslice5 50
Got my SA credential
Found the slice, asking for a credential ...
Got the slice credential, renewing the slice at the SA ...
Renewed the slice, asking for slice credential again
Got the slice credential, renewing the sliver
Sliver has been renewed until 20100718T16:03:14
sshalini@ubuntu:~/protogeni/test$
```

Fig. 4.19: Renewing the sliver for a specific time from initial time settings of sliver.

DeleteSliver: deletesliver.py deletes a sliver specified in the command. If a sliver has been renewed, deletesliver.py deletes the sliver and returns the slice credentials. Here we tried to *redeemticket.py* just after deleting the sliver and it did it successfully so sliver was again active.



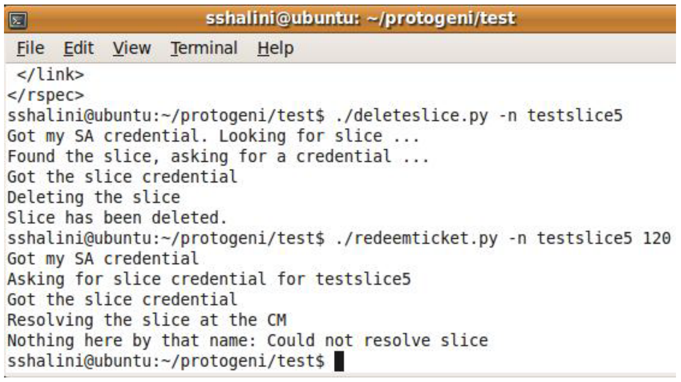
sshalini@ubuntu: ~/protogeni/test
File Edit View Terminal Help

```
Sliver has been renewed until 20100718T16:03:14
sshalini@ubuntu:~/protogeni/test$ ./deletesliver.py -n testslice5
Got my SA credential. Looking for slice ...
Found the slice, asking for a credential ...
Got the slice credential, asking for a sliver credential ...
Resolving the slice at the CM
{'urn': 'urn:publicid:IDN+emulab.net+slice+testslice5', 'sliver_urn': 'urn:pul
licid:IDN+emulab.net+sliver+14301'}
Got the sliver credential, deleting the sliver
Sliver has been deleted. Ticket for remaining time:
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
```

```
sshalini@ubuntu:~/protogeni/test$ ./redeemticket.py -n testslice5 120
Got my SA credential
Asking for slice credential for testslice5
Got the slice credential
Resolving the slice at the CM
{'urn': 'urn:publicid:IDN+emulab.net+slice+testslice5', 'ticket_urn': 'urn:pub
licid:IDN+emulab.net+ticket+59143'}
Asking for a copy of the ticket
Got the ticket
Asking for sliver credential
No slice or aggregate here:Redeeming the ticket
```

Fig. 4.20: Deletion of sliver and redemption of ticket that made it active again

DeleteSlice: deleteslice.py deletes all the details at CM and releases the resources with specified slice. An empty slice can be existed if resources are already been released through deletesliver.py.


sshalini@ubuntu: ~/protogeni/test
File Edit View Terminal Help

```
 </link>
</rspec>
sshalini@ubuntu:~/protogeni/test$ ./deleteslice.py -n testslice5
Got my SA credential. Looking for slice ...
Found the slice, asking for a credential ...
Got the slice credential
Deleting the slice
Slice has been deleted.
sshalini@ubuntu:~/protogeni/test$ ./redeemticket.py -n testslice5 120
Got my SA credential
Asking for slice credential for testslice5
Got the slice credential
Resolving the slice at the CM
Nothing here by that name: Could not resolve slice
sshalini@ubuntu:~/protogeni/test$ 
```

Fig. 4.21: Deletion of slice and no further slice record

UpdateSliver: This test script, updatesliver.py is supposed to modify the sliver resources without deleting the sliver. In trial, the sliver is being updated in records and returns the details on screen with information to redeem before a particular time but sliver status was showing the same details as before applying the updatesliver.py. The updatesliver.py command was repeated with a different RSpec file for a single node but it could not create geniobject, and sliver could not be started or revived further. Updating a sliver is a bit complex as it involves quick redemption of new ticket. A detailed description of outputs is attached as Annexure A. Alternate temporary solution may be to create a new slice/sliver other than updating the existing sliver.

Special operations

list-ch.py, *listusage.py*, *unregisterslice.py*, and *shutdownslice.py* are privileged operations which normally can be invoked by the clearinghouse only.

Advanced operations

TunTest: *tuntest.py* can create two slivers with two different Component Managers under a single slice.

./tuntest utahemulab ukgeni

Utahemulab is Emulab of Utah and ukgeni is Emulab facility of Kentucky.

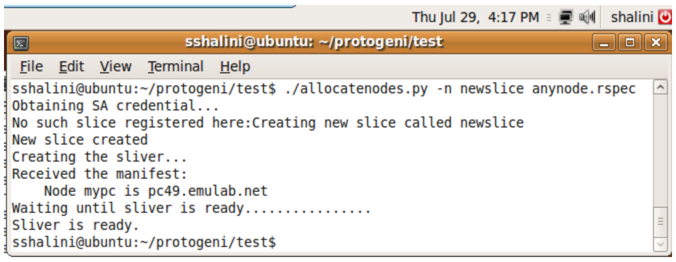
Additional Test Scripts

ProtoGENI test scripts are initial scripts to use ProtoGENI resources and to see the basic GENI functionality. Some test scripts are modified by Utah to make them more user friendly and to show the sliver and node status in a cleaner way with necessary information only. These test scripts are added as gec8tutorial-scripts.tar.gz recently by Utah team [54].

1) Allocatenodes.py

This is a similar test script to createsliver.py which allocates resources to the sliver as specified in RSpec file. Output provides the allotted nodes to the sliver and also returns the status of the sliver.

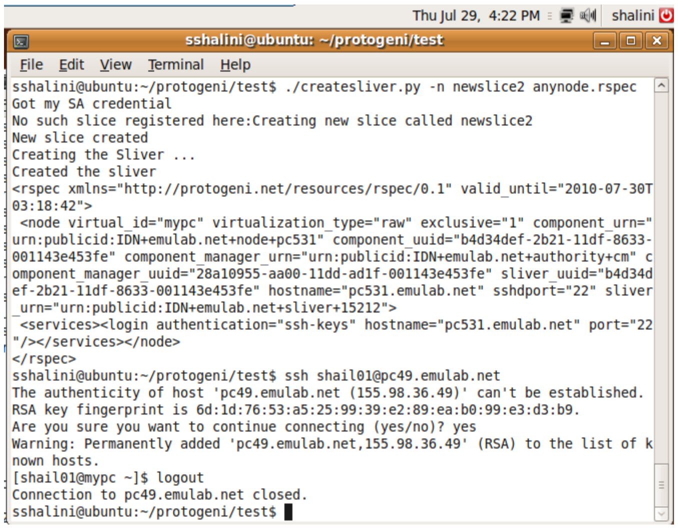
./allocatenodes.py –n newslice1 anynode.rspec


Thu Jul 29, 4:17 PM
shalini
sshalini@ubuntu: ~/protogeni/test
File Edit View Terminal Help

```
sshalini@ubuntu:~/protogeni/test$ ./allocatenodes.py -n newslice anynode.rspec
Obtaining SA credential...
No such slice registered here:Creating new slice called newslice
New slice created
Creating the sliver...
Received the manifest:
    Node mypc is pc49.emulab.net
Waiting until sliver is ready...............
Sliver is ready.
sshalini@ubuntu:~/protogeni/test$
```

Fig. 4.22: Test Script allocatenodes.py

Till now we were only using .xml RSpec files to request resources but following this example, we also created sliver through createsliver.py with .rspec file as shown in fig 4.23.


Thu Jul 29, 4:22 PM
shalini
sshalini@ubuntu: ~/protogeni/test
File Edit View Terminal Help

```
sshalini@ubuntu:~/protogeni/test$ ./createsliver.py -n newslice2 anynode.rspec
Got my SA credential
No such slice registered here:Creating new slice called newslice2
New slice created
Creating the Sliver ...
Created the sliver
<rspec xmlns="http://protogeni.net/resources/rspec/0.1" valid_until="2010-07-30T
03:18:42">
 <node virtual_id="mypc" virtualization_type="raw" exclusive="1" component_urn="
urn:publicid:IDN+emulab.net+node+pc531" component_uuid="b4d34def-2b21-11df-8633-
001143e453fe" component_manager_urn="urn:publicid:IDN+emulab.net+authority+cm" c
omponent_manager_uuid="28a10955-aa00-11dd-ad1f-001143e453fe" sliver_uuid="b4d34d
ef-2b21-11df-8633-001143e453fe" hostname="pc531.emulab.net" sshdport="22" sliver
_urn="urn:publicid:IDN+emulab.net+sliver+15212">
 <services><login authentication="ssh-keys" hostname="pc531.emulab.net" port="22
"/></services></node>
</rspec>
sshalini@ubuntu:~/protogeni/test$ ssh shail01@pc49.emulab.net
The authenticity of host 'pc49.emulab.net (155.98.36.49)' can't be established.
RSA key fingerprint is 6d:1d:76:53:a5:25:99:39:e2:89:ea:b0:99:e3:d3:b9.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'pc49.emulab.net,155.98.36.49' (RSA) to the list of k
nown hosts.
[shail01@mypc ~]$ logout
Connection to pc49.emulab.net closed.
sshalini@ubuntu:~/protogeni/test$ █
```

Fig. 4.23: createsliver.py with anynode.rspec RSpec file

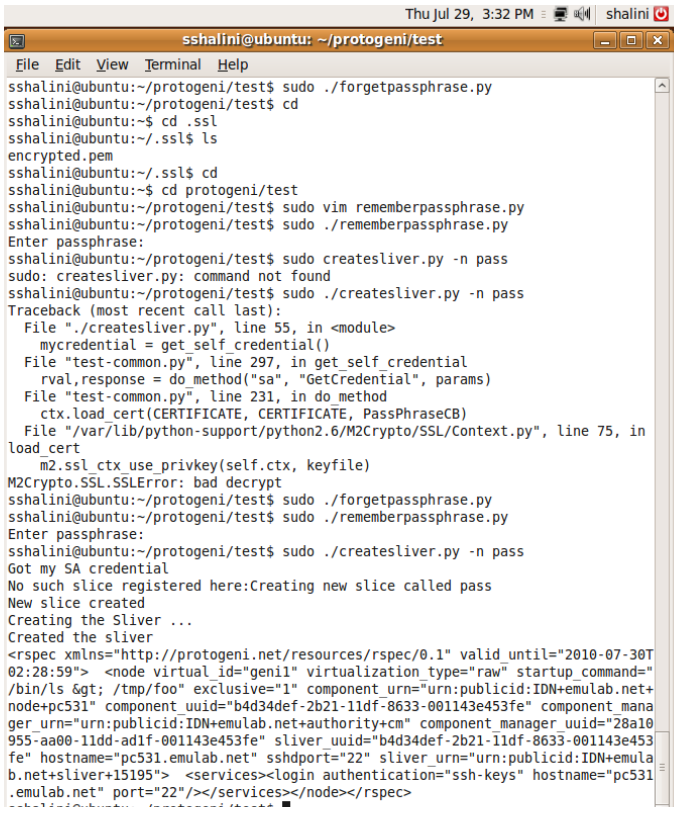
2) *Rememberpassphrase.py and forgetpassphrase.py*

These two test scripts provide a convenient way to enhance the security of physical machine. SSH keys are protected with a passphrase and this passphrase can be stored at a specific place to avoid retyping of it again and again but this also makes physical machine a bit prone to attack as attacker may know the particular file to look for passphrase. Rememberpassphrase.py enables users to type the correct passphrase in command line and creates the required file at required place to work with ProtoGENI settings.

As per fig. 4.24, forgetpassphrase.py deletes the passphrase file and rememberpassphrase.py creates a passphrase file at proper place. Rememberpassphrase.py creates the file but if you enter a wrong passphrase, system will not accept it to create a sliver. Once a correct passphrase is provided, system is good to be in action.

3) *Showuser.py*

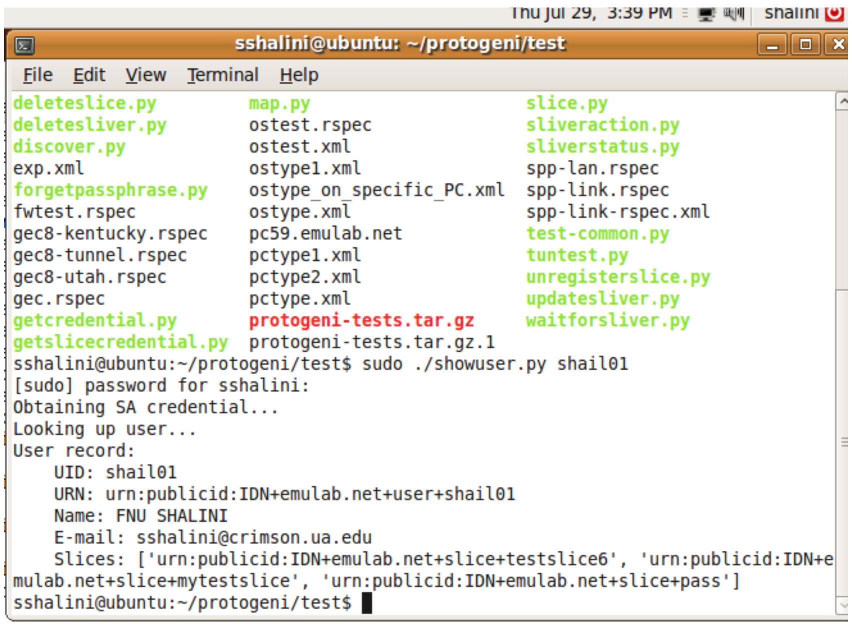
This test script is to get information about active slices by a particular user. It shows the slices which are not expired. So a slice/sliver may be deleted already but still will be shown in output of showuser.py if it is not expired at clearinghouse.


Thu Jul 29, 3:32 PM
shalini
sshalini@ubuntu: ~/protogeni/test
File Edit View Terminal Help

```
sshalini@ubuntu:~/protogeni/test$ sudo ./forgetpassphrase.py
sshalini@ubuntu:~/protogeni/test$ cd
sshalini@ubuntu:~$ cd .ssl
sshalini@ubuntu:~/.ssl$ ls
encrypted.pem
sshalini@ubuntu:~/.ssl$ cd
sshalini@ubuntu:~$ cd protogeni/test
sshalini@ubuntu:~/protogeni/test$ sudo vim rememberpassphrase.py
sshalini@ubuntu:~/protogeni/test$ sudo ./rememberpassphrase.py
Enter passphrase:
sshalini@ubuntu:~/protogeni/test$ sudo createsliver.py -n pass
sudo: createsliver.py: command not found
sshalini@ubuntu:~/protogeni/test$ sudo ./createsliver.py -n pass
Traceback (most recent call last):
  File "./createsliver.py", line 55, in <module>
    mycredential = get_self_credential()
  File "test-common.py", line 297, in get_self_credential
    rval,response = do_method("sa", "GetCredential", params)
  File "test-common.py", line 231, in do_method
    ctx.load_cert(CERTIFICATE, CERTIFICATE, PassPhraseCB)
  File "/var/lib/python-support/python2.6/M2Crypto/SSL/Context.py", line 75, in
load_cert
    m2.ssl_ctx_use_privkey(self.ctx, keyfile)
M2Crypto.SSL.SSLError: bad decrypt
sshalini@ubuntu:~/protogeni/test$ sudo ./forgetpassphrase.py
sshalini@ubuntu:~/protogeni/test$ sudo ./rememberpassphrase.py
Enter passphrase:
sshalini@ubuntu:~/protogeni/test$ sudo ./createsliver.py -n pass
Got my SA credential
No such slice registered here:Creating new slice called pass
New slice created
Creating the Sliver ...
Created the sliver
<rspec xmlns="http://protogeni.net/resources/rspec/0.1" valid_until="2010-07-30T
02:28:59">  <node virtual_id="geni1" virtualization_type="raw" startup_command="
/bin/ls &gt; /tmp/foo" exclusive="1" component_urn="urn:publicid:IDN+emulab.net+
node+pc531" component_uuid="b4d34def-2b21-11df-8633-001143e453fe" component_mana
ger_urn="urn:publicid:IDN+emulab.net+authority+cm" component_manager_uuid="28a10
955-aa00-11dd-ad1f-001143e453fe" sliver_uuid="b4d34def-2b21-11df-8633-001143e453
fe" hostname="pc531.emulab.net" sshdport="22" sliver_urn="urn:publicid:IDN+emula
b.net+sliver+15195">  <services><login authentication="ssh-keys" hostname="pc531
.emulab.net" port="22"/></services></node></rspec>
```

Fig. 4.24: Passphrase related test scripts

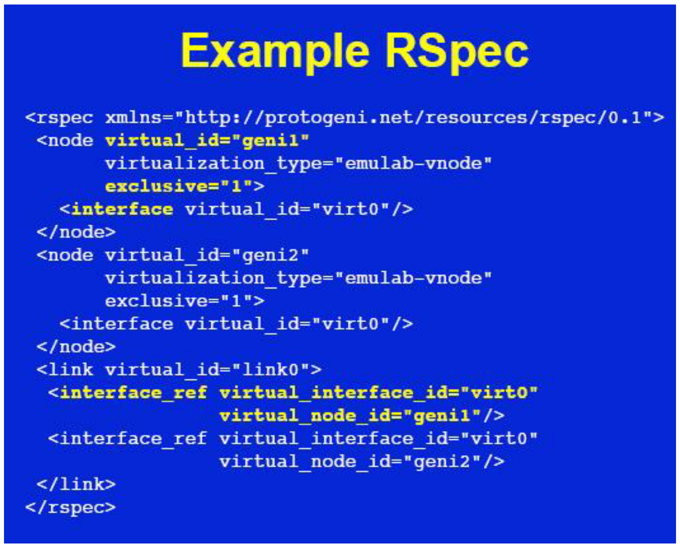
User can use this information to check the status of slivers and can delete the slivers and releasing the resources if not using them.


Thu Jul 29, 3:39 PM
shalini
sshalini@ubuntu: ~/protogeni/test
File Edit View Terminal Help

```
deleteslice.py          map.py                     slice.py
deletesliver.py         ostest.rspec               sliveraction.py
discover.py             ostest.xml                 sliverstatus.py
exp.xml                 ostype1.xml                spp-lan.rspec
forgetpassphrase.py     ostype_on_specific_PC.xml  spp-link.rspec
fwtest.rspec            ostype.xml                 spp-link-rspec.xml
gec8-kentucky.rspec     pc59.emulab.net            test-common.py
gec8-tunnel.rspec       pctype1.xml                tuntest.py
gec8-utah.rspec         pctype2.xml                unregisterslice.py
gec.rspec               pctype.xml                 updatesliver.py
getcredential.py        protogeni-tests.tar.gz     waitforsliver.py
getslicecredential.py   protogeni-tests.tar.gz.1
sshalini@ubuntu:~/protogeni/test$ sudo ./showuser.py shail01
[sudo] password for sshalini:
Obtaining SA credential...
Looking up user...
User record:
    UID: shail01
    URN: urn:publicid:IDN+emulab.net+user+shail01
    Name: FNU SHALINI
    E-mail: sshalini@crimson.ua.edu
    Slices: ['urn:publicid:IDN+emulab.net+slice+testslice6', 'urn:publicid:IDN+e
mulab.net+slice+mytestslice', 'urn:publicid:IDN+emulab.net+slice+pass']
sshalini@ubuntu:~/protogeni/test$ 
```

Fig. 4.25: Execution of showuser.py test script

RSpec .xml file can be borrowed from tutorial of ProtoGENI website [28] and it works fine to check initial functionality.



Example RSpec

```
<rspec xmlns="http://protogeni.net/resources/rspec/0.1">
 <node virtual_id="geni1"
       virtualization_type="emulab-vnode"
       exclusive="1">
   <interface virtual_id="virt0"/>
 </node>
 <node virtual_id="geni2"
       virtualization_type="emulab-vnode"
       exclusive="1">
   <interface virtual_id="virt0"/>
 </node>
 <link virtual_id="link0">
  <interface_ref virtual_interface_id="virt0"
                 virtual_node_id="geni1"/>
  <interface_ref virtual_interface_id="virt0"
                 virtual_node_id="geni2"/>
 </link>
</rspec>
```

Fig. 4.26: Example RSpec can be used as myrspec.xml [28]

If more resources are required, same RSpec .xml file can be modified or a new one can be created once the user is more familiar with RSpec details. OS type of a particular node and a particular kind of PC can be defined in RSpec .xml file.

Users can join Geni-user mail group to get over difficulties and can share questions and findings with all other researchers.

We run and verified the functioning of test scripts available and observed the associated problems which will be discussed in details in result and analysis section. Experiments helped to distinguish between different ways to acquire resources and preferred the simpler way to acquiring and releasing the ProtoGENI resources to ensure a better accessibility and availability of resources to all ProtoGENI users. An alternate solution for “wrong host” problem was also suggested which worked fine.

ProtoGENI Resources: Accessibility and Availability through RSpecs

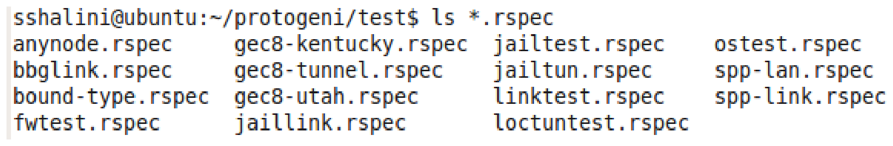
RSpec- Resource Specifications: The ProtoGENI RSpec is ProtoGENI’s mechanism for advertising, requesting, and describing the resources used by experimenters. RSpecs interact with nearly every piece of the GENI architecture. ProtoGENI entities create and modify different kinds of RSpecs during different operations to design and implement an experiment.

Requests specify which resources a client is selecting from Component Managers. Request RSpecs may contain a complete, partial or empty mapping between physical components and abstract nodes and links [42]. If there is no mapping, a request is known as unbound. A bound request contains mapped links and nodes.

Ticket operations are alternate to acquire a sliver and resources. Component Manager returns a ticket or a failure to complete or reject the request of resources.

Usage of RSpecs in Requesting Resources

There are many RSpec files [Fig.27] available with test script suite which can be downloaded from ProtoGENI web link [28].

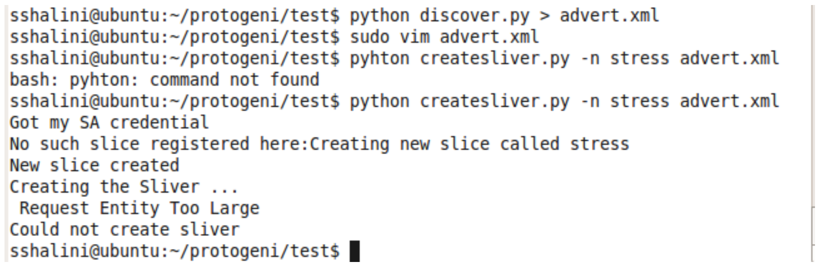


```
sshalini@ubuntu:~/protogeni/test$ ls *.rspec
anynode.rspec     gec8-kentucky.rspec  jailtest.rspec    ostest.rspec
bbglink.rspec     gec8-tunnel.rspec    jailtun.rspec     spp-lan.rspec
bound-type.rspec  gec8-utah.rspec      linktest.rspec    spp-link.rspec
fwtest.rspec      jaillink.rspec       loctuntest.rspec
```

Fig. 4.27: .rspec files are available in ProtoGENI test scripts suite

All resources in one sliver, possible?

All the available resources can be known through discover.py and the output can be saved in advert.xml file. We tried to acquire all the resources all together by putting this advert.xml to create a sliver as it could be a potential threat if all resources can be acquired by one user in a single sliver.



```
sshalini@ubuntu:~/protogeni/test$ python discover.py > advert.xml
sshalini@ubuntu:~/protogeni/test$ sudo vim advert.xml
sshalini@ubuntu:~/protogeni/test$ pyhton createsliver.py -n stress advert.xml
bash: pyhton: command not found
sshalini@ubuntu:~/protogeni/test$ python createsliver.py -n stress advert.xml
Got my SA credential
No such slice registered here:Creating new slice called stress
New slice created
Creating the Sliver ...
 Request Entity Too Large
Could not create sliver
sshalini@ubuntu:~/protogeni/test$
```

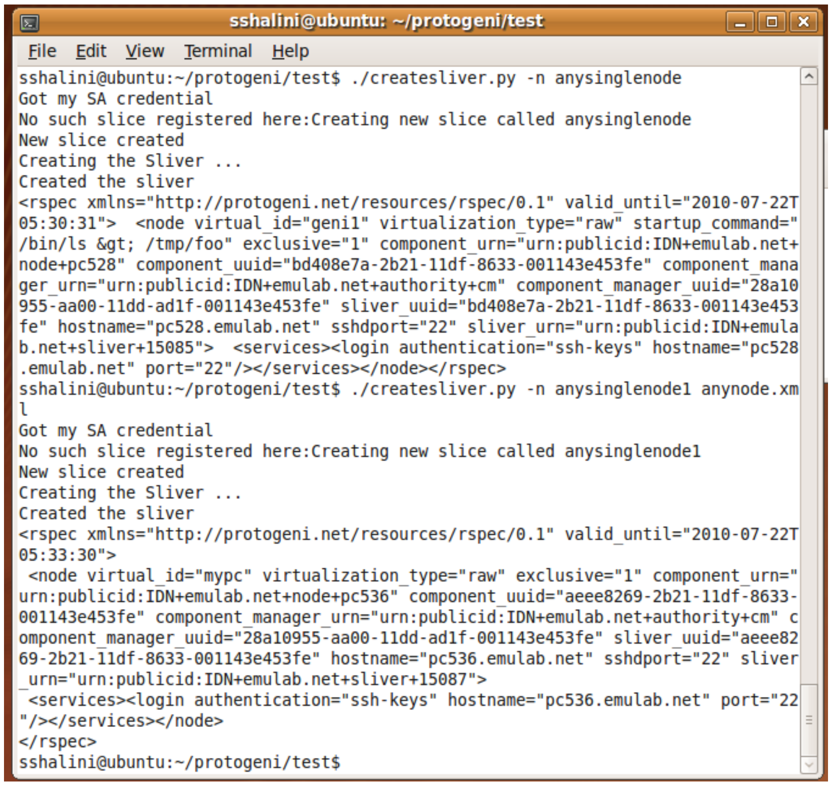
Fig. 4.28: Effort to acquire all available resources in one sliver

From fig. 4.28 we can see that system doesn’t allow the creation of a single sliver having all available resources in that one sliver.

System is taking care of this issue and any abnormal demand of resources can be monitored by the ProtoGENI authorities and a sliver can be terminated if required.

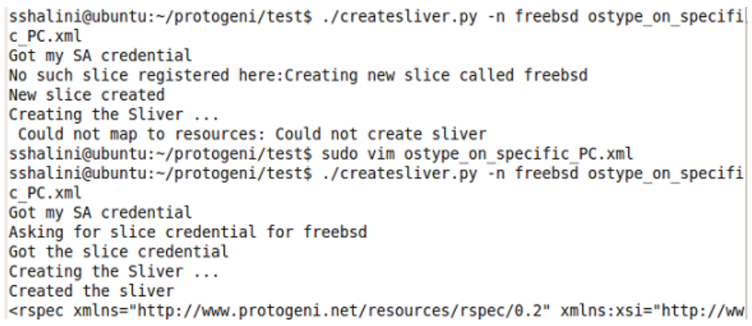
Simplest RSpec: for any available single node

A single node can be obtained without any given specification without providing any RSpec .xml file or with anynode.xml [See Appendix]


sshalini@ubuntu: ~/protogeni/test
File Edit View Terminal Help

```
sshalini@ubuntu:~/protogeni/test$ ./createsliver.py -n anysinglenode
Got my SA credential
No such slice registered here:Creating new slice called anysinglenode
New slice created
Creating the Sliver ...
Created the sliver
<rspec xmlns="http://protogeni.net/resources/rspec/0.1" valid_until="2010-07-22T
05:30:31">  <node virtual_id="geni1" virtualization_type="raw" startup_command="
/bin/ls &gt; /tmp/foo" exclusive="1" component_urn="urn:publicid:IDN+emulab.net+
node+pc528" component_uuid="bd408e7a-2b21-11df-8633-001143e453fe" component_mana
ger_urn="urn:publicid:IDN+emulab.net+authority+cm" component_manager_uuid="28a10
955-aa00-11dd-ad1f-001143e453fe" sliver_uuid="bd408e7a-2b21-11df-8633-001143e453
fe" hostname="pc528.emulab.net" sshdport="22" sliver_urn="urn:publicid:IDN+emula
b.net+sliver+15085">  <services><login authentication="ssh-keys" hostname="pc528
.emulab.net" port="22"/></services></node></rspec>
sshalini@ubuntu:~/protogeni/test$ ./createsliver.py -n anysinglenode1 anynode.xm
l
Got my SA credential
No such slice registered here:Creating new slice called anysinglenode1
New slice created
Creating the Sliver ...
Created the sliver
<rspec xmlns="http://protogeni.net/resources/rspec/0.1" valid_until="2010-07-22T
05:33:30">
 <node virtual_id="mypc" virtualization_type="raw" exclusive="1" component_urn="
urn:publicid:IDN+emulab.net+node+pc536" component_uuid="aeee8269-2b21-11df-8633-
001143e453fe" component_manager_urn="urn:publicid:IDN+emulab.net+authority+cm" c
omponent_manager_uuid="28a10955-aa00-11dd-ad1f-001143e453fe" sliver_uuid="aeee82
69-2b21-11df-8633-001143e453fe" hostname="pc536.emulab.net" sshdport="22" sliver
_urn="urn:publicid:IDN+emulab.net+sliver+15087">
 <services><login authentication="ssh-keys" hostname="pc536.emulab.net" port="22
"/></services></node>
</rspec>
sshalini@ubuntu:~/protogeni/test$
```

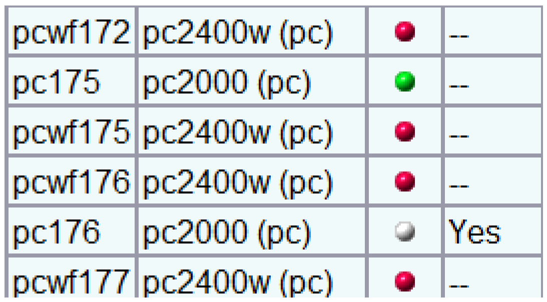
Fig. 4.29: Requesting a not-specific single node with and without RSpec file



```
sshalini@ubuntu:~/protogeni/test$ ./createsliver.py -n freebsd ostype_on_specifi
c_PC.xml
Got my SA credential
No such slice registered here:Creating new slice called freebsd
New slice created
Creating the Sliver ...
 Could not map to resources: Could not create sliver
sshalini@ubuntu:~/protogeni/test$ sudo vim ostype_on_specific_PC.xml
sshalini@ubuntu:~/protogeni/test$ ./createsliver.py -n freebsd ostype_on_specifi
c_PC.xml
Got my SA credential
Asking for slice credential for freebsd
Got the slice credential
Creating the Sliver ...
Created the sliver
<rspec xmlns="http://www.protogeni.net/resources/rspec/0.2" xmlns:xsi="http://ww
```

Fig. 4.30: Non-availability of a particular resource requested in RSpec file.

Sometimes a particular resource requested in RSpec may not be available so, it is a good idea to verify the node status through Emulab account.



pcwf172	pc2400w (pc)	🔴	--
pc175	pc2000 (pc)	🟢	--
pcwf175	pc2400w (pc)	🔴	--
pcwf176	pc2400w (pc)	🔴	--
pc176	pc2000 (pc)	⚪	Yes
pcwf177	pc2400w (pc)	🔴	--

Fig. 4.31: Requested PC was busy- Status can be seen at Emulab

Rspec in fig 26 is easy to understand and the most utilized RSpec file in our experiments. It provides two GENI nodes without any particular node type, ID or OS which results in two available up nodes with default linux OS image (as on 07-21-2010).

Different RSpecs files were created to see if system can provide the requested resources as per specified or not.

All other available RSpec files were observed for what details they are providing to help in requesting resources with proper definitions.

Software installation on remote nodes through RSpec file

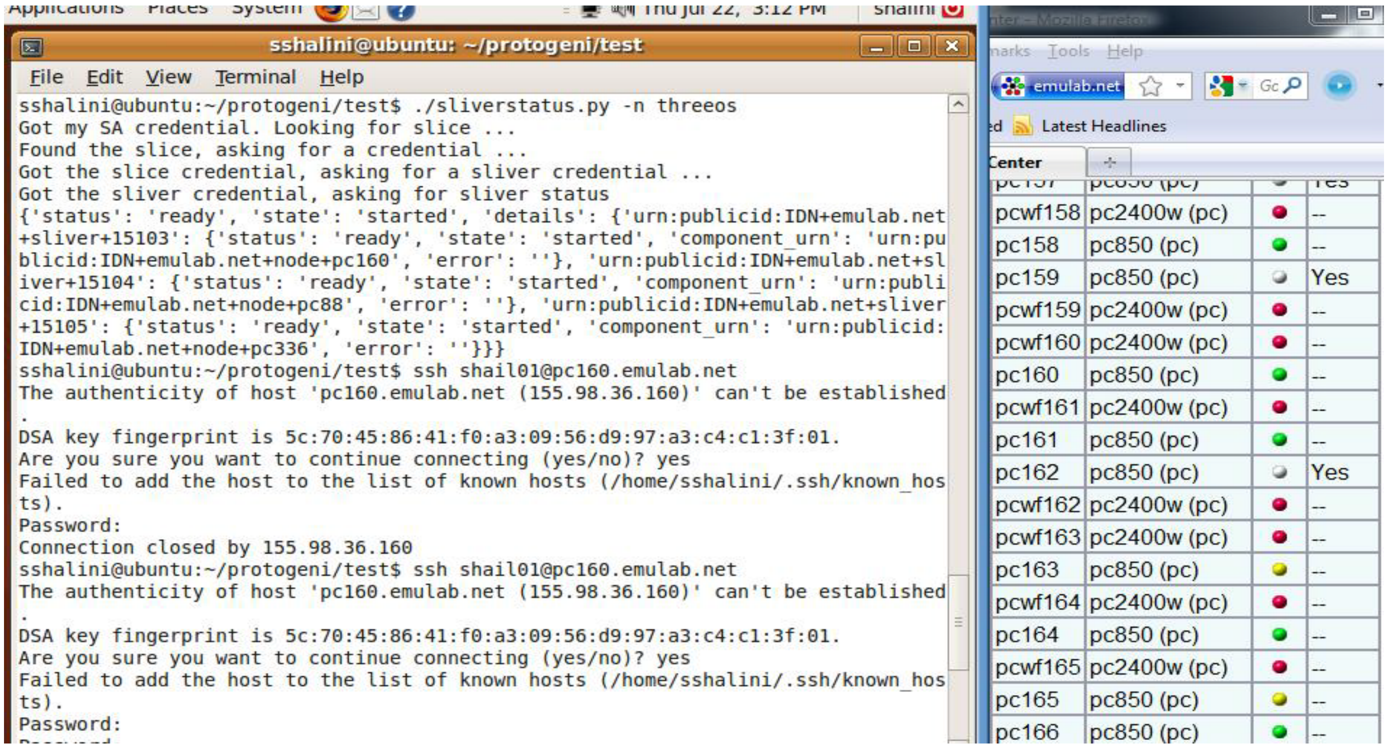
RSpec can be used to download and unpack tar files from web link to remote node. remoteexecution.xml is generated as per web example. The example was modified to get a valid tar file from web link.

Slice *remoteexe* was created with RSpec file. ProtoGENI test script suite was supposed to be downloaded and being unpacked on remote node /local folder which was specified in RSpec. Sliver was created successfully and /local folder was verified to see the existence of test script suite files. This was done successfully, so tar files can be downloaded on remote node.

Table-1 gives a brief description of each available RSpec with ProtoGENI and of some other RSpecs developed to see specific functionality.

S.no	**RSpec file**	**Brief Description**	**Source**
1	anynode	One node- no other specification	Test scripts suite
2	fwtest	Asking for firewall	-do-
3	jaillink	Two nodes, one link- no specific definition of resources	-do-
4	jailtest	Two nodes, no link	-do-
5	jailtun	Two nodes, one link, one tunnel	-do-
6	gec8-kentucky	One node from Kentucky CM	-do-
7	linktest	Two nodes, one link	-do-
8	loctuntest	Two nodes with particular OS, link type as tunnel	-do-
9	ostest	One node with a particular OS	-do-
10	Spp-lan	3 nodes, link specification bandwidth etc., lan-3 interfaces	-do-
11	Spp-link	2 nodes, one link with specifications	-do-
12	Gec8-tunnel	Link definitions for creating tunnel between two nodes at two different CMs	-do-
13	Bound-type	One particular PC type	-do-
14	gec8-utah	3 nodes, OS-FEDORA8-OVZ-STD, 2 links	-do-
15	bbglink	2 bbg nodes, one link with specifications	-do-
16	ostype3	One node-specific node_id and a particular OS	Geni-user mail
17	sharednode	Two nodes type pcvm, one link with specifications	ProtoGENI web details
18	remoteexecution	To download and unpack the tar files or remote node	-do-

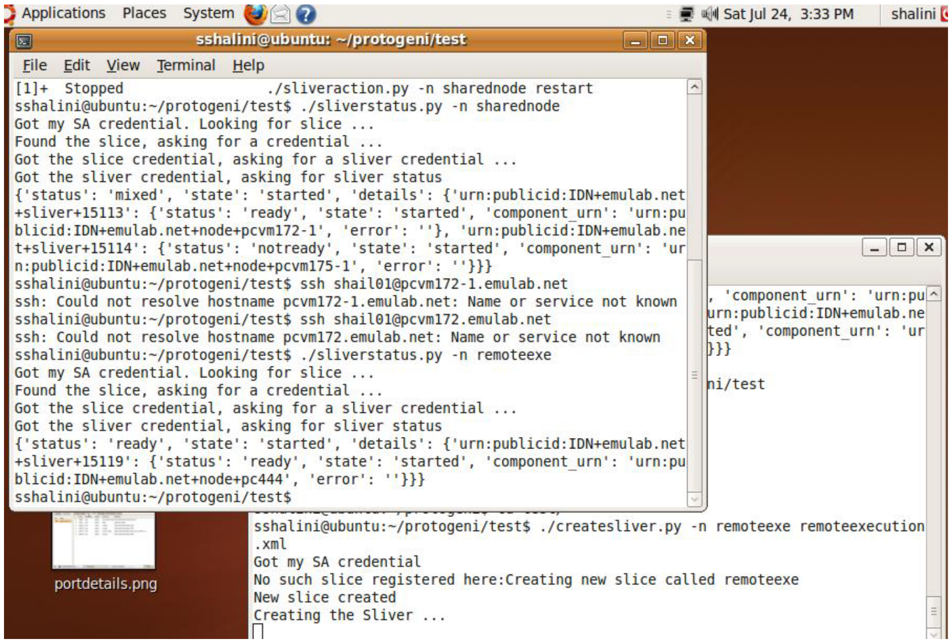
Table-4.1: A brief description of helping RSpec examples



```
sshalini@ubuntu:~/protogeni/test$ ./sliverstatus.py -n threeos
Got my SA credential. Looking for slice ...
Found the slice, asking for a credential ...
Got the slice credential, asking for a sliver credential ...
Got the sliver credential, asking for sliver status
{'status': 'ready', 'state': 'started', 'details': {'urn:publicid:IDN+emulab.net
+sliver+15103': {'status': 'ready', 'state': 'started', 'component_urn': 'urn:pu
blicid:IDN+emulab.net+node+pc160', 'error': ''}, 'urn:publicid:IDN+emulab.net+sl
iver+15104': {'status': 'ready', 'state': 'started', 'component_urn': 'urn:publi
cid:IDN+emulab.net+node+pc88', 'error': ''}, 'urn:publicid:IDN+emulab.net+sliver
+15105': {'status': 'ready', 'state': 'started', 'component_urn': 'urn:publicid:
IDN+emulab.net+node+pc336', 'error': ''}}}
sshalini@ubuntu:~/protogeni/test$ ssh shail01@pc160.emulab.net
The authenticity of host 'pc160.emulab.net (155.98.36.160)' can't be established
.
DSA key fingerprint is 5c:70:45:86:41:f0:a3:09:56:d9:97:a3:c4:c1:3f:01.
Are you sure you want to continue connecting (yes/no)? yes
Failed to add the host to the list of known hosts (/home/sshalini/.ssh/known_hos
ts).
Password:
Connection closed by 155.98.36.160
sshalini@ubuntu:~/protogeni/test$ ssh shail01@pc160.emulab.net
The authenticity of host 'pc160.emulab.net (155.98.36.160)' can't be established
.
DSA key fingerprint is 5c:70:45:86:41:f0:a3:09:56:d9:97:a3:c4:c1:3f:01.
Are you sure you want to continue connecting (yes/no)? yes
Failed to add the host to the list of known hosts (/home/sshalini/.ssh/known_hos
ts).
Password:
```

pcwf158	pc2400w (pc)	🔴	--
pc158	pc850 (pc)	🟢	--
pc159	pc850 (pc)	⚪	Yes
pcwf159	pc2400w (pc)	🔴	--
pcwf160	pc2400w (pc)	🔴	--
pc160	pc850 (pc)	🟢	--
pcwf161	pc2400w (pc)	🔴	--
pc161	pc850 (pc)	🟢	--
pc162	pc850 (pc)	⚪	Yes
pcwf162	pc2400w (pc)	🔴	--
pcwf163	pc2400w (pc)	🔴	--
pc163	pc850 (pc)	🟡	--
pcwf164	pc2400w (pc)	🔴	--
pc164	pc850 (pc)	🟢	--
pcwf165	pc2400w (pc)	🔴	--
pc165	pc850 (pc)	🟡	--
pc166	pc850 (pc)	🟢	--

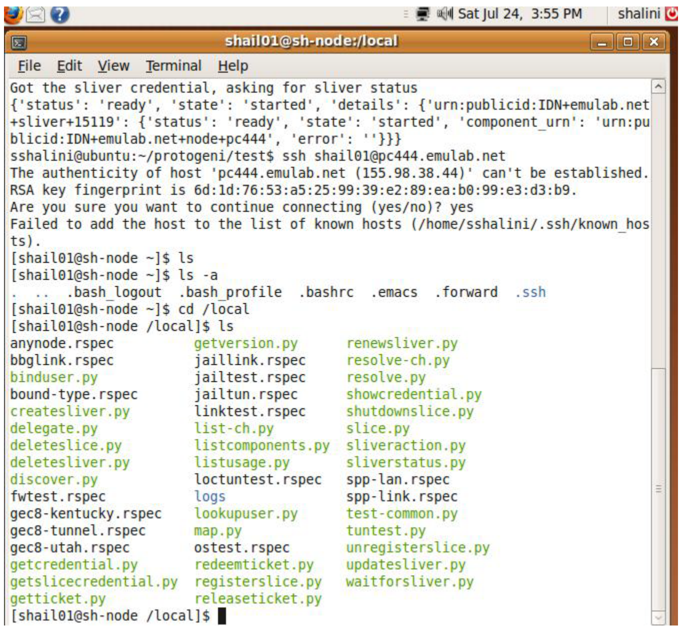
Fig. 4.32: Node status of slice/sliver *threeos* through sliverstatus.py and in Emulab as good but could not get control on pc160



```
[1]+  Stopped                 ./sliveraction.py -n sharednode restart
sshalini@ubuntu:~/protogeni/test$ ./sliverstatus.py -n sharednode
Got my SA credential. Looking for slice ...
Found the slice, asking for a credential ...
Got the slice credential, asking for a sliver credential ...
Got the sliver credential, asking for sliver status
{'status': 'mixed', 'state': 'started', 'details': {'urn:publicid:IDN+emulab.net
+sliver+15113': {'status': 'ready', 'state': 'started', 'component_urn': 'urn:pu
blicid:IDN+emulab.net+node+pcvm172-1', 'error': ''}, 'urn:publicid:IDN+emulab.ne
t+sliver+15114': {'status': 'notready', 'state': 'started', 'component_urn': 'ur
n:publicid:IDN+emulab.net+node+pcvm175-1', 'error': ''}}}
sshalini@ubuntu:~/protogeni/test$ ssh shail01@pcvm172-1.emulab.net
ssh: Could not resolve hostname pcvm172-1.emulab.net: Name or service not known
sshalini@ubuntu:~/protogeni/test$ ssh shail01@pcvm172.emulab.net
ssh: Could not resolve hostname pcvm172.emulab.net: Name or service not known
sshalini@ubuntu:~/protogeni/test$ ./sliverstatus.py -n remoteexe
Got my SA credential. Looking for slice ...
Found the slice, asking for a credential ...
Got the slice credential, asking for a sliver credential ...
Got the sliver credential, asking for sliver status
{'status': 'ready', 'state': 'started', 'details': {'urn:publicid:IDN+emulab.net
+sliver+15119': {'status': 'ready', 'state': 'started', 'component_urn': 'urn:pu
blicid:IDN+emulab.net+node+pc444', 'error': ''}}}
sshalini@ubuntu:~/protogeni/test$
```

```
sshalini@ubuntu:~/protogeni/test$ ./createsliver.py -n remoteexe remoteexecution
.xml
Got my SA credential
No such slice registered here:Creating new slice called remoteexe
New slice created
Creating the Sliver ...
```

Fig. 4.33: sliver creation to download software at remote node



```
Got the sliver credential, asking for sliver status
{'status': 'ready', 'state': 'started', 'details': {'urn:publicid:IDN+emulab.net
+sliver+15119': {'status': 'ready', 'state': 'started', 'component_urn': 'urn:pu
blicid:IDN+emulab.net+node+pc444', 'error': ''}}}
sshalini@ubuntu:~/protogeni/test$ ssh shail01@pc444.emulab.net
The authenticity of host 'pc444.emulab.net (155.98.38.44)' can't be established.
RSA key fingerprint is 6d:1d:76:53:a5:25:99:39:e2:89:ea:b0:99:e3:d3:b9.
Are you sure you want to continue connecting (yes/no)? yes
Failed to add the host to the list of known hosts (/home/sshalini/.ssh/known_hos
ts).
[shail01@sh-node ~]$ ls
[shail01@sh-node ~]$ ls -a
.  ..  .bash_logout  .bash_profile  .bashrc  .emacs  .forward  .ssh
[shail01@sh-node ~]$ cd /local
[shail01@sh-node /local]$ ls
anynode.rspec          getversion.py      renewsliver.py
bbglink.rspec          jaillink.rspec     resolve-ch.py
binduser.py            jailtest.rspec     resolve.py
bound-type.rspec       jailtun.rspec      showcredential.py
createsliver.py        linktest.rspec     shutdownslice.py
delegate.py            list-ch.py         slice.py
deleteslice.py         listcomponents.py  sliveraction.py
deletesliver.py        listusage.py       sliverstatus.py
discover.py            loctuntest.rspec   spp-lan.rspec
fwtest.rspec           logs               spp-link.rspec
gec8-kentucky.rspec    lookupuser.py      test-common.py
gec8-tunnel.rspec      map.py             tuntest.py
gec8-utah.rspec        ostest.rspec       unregisterslice.py
getcredential.py       redeemticket.py    updatesliver.py
getslicecredential.py  registerslice.py   waitforsliver.py
getticket.py           releaseticket.py
[shail01@sh-node /local]$ █
```

Fig. 4.34: tar file downloaded and unpacked at remote node

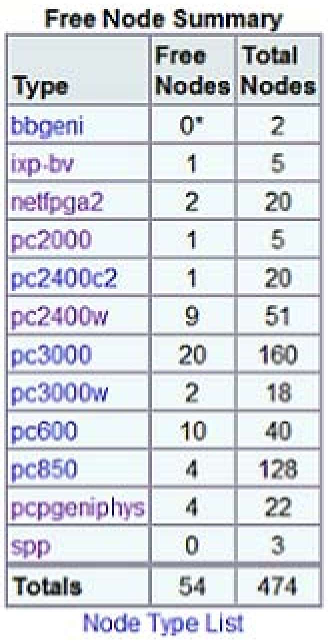
We tried few commands at remote node but ProtoGENI test scripts don't work as they need supporting software like m2crypto and python installed at that node as primary things to move further. More than one tar file can be installed install on remote node. Rspec as remtwo.xml can install a list of software separated by a semicolon as directed at [57].

RSpecs are modified to see the variations and execution of RSpec examples. These details provide a simplification about available RSpec examples, their implementation, and their limitations in current scenario. More experiments will explore further advancements in Rspec file and their intended usage.

Emulab Resources: Emulab contains a vast array of resources including wired cluster PCs that can run essentially arbitrary operating systems including Windows, Linux, and FreeBSD; physically distributed 802.11 wireless PCs; network processors; sensor boards; and software radio platforms. There are total 474 nodes distributed among 12 node types [9].

Each node type has given an identity and a set of its characteristics are available on Emulab user account as node type information including node type, node class, default_osid, processor, and virtnode_capacity. Node Control Center provides free node summary which can help experimenter to design their experiments accordingly. Users also can request other users to free the desired nodes through emailing to geni-users email group.

Emulab resources are allocated to a single user at a time. If nodes are being used by some experimenters, it may hinder the possibility of other experiments to run because of unavailability of resources. Interaction between experimenter and ProtoGENI control framework is required to conduct the network experiments.



Free Node Summary

Type	Free Nodes	Total Nodes
bbgeni	0*	2
ixp-bv	1	5
netfpga2	2	20
pc2000	1	5
pc2400c2	1	20
pc2400w	9	51
pc3000	20	160
pc3000w	2	18
pc600	10	40
pc850	4	128
pcpgeniphys	4	22
spp	0	3
Totals	54	474

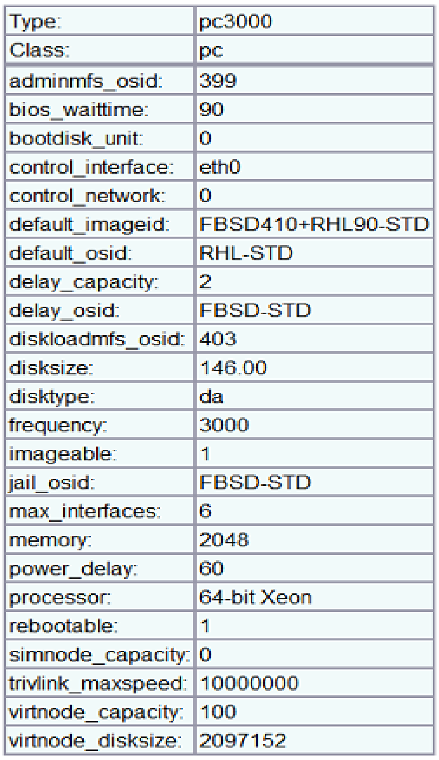
Node Type List

Fig. 4.35: Emulab Resources: Free Node Summary on Emulab site [9]

Experimenter can define their desired types of nodes and also can select the operating system. ProtoGENI allows users to experiment on PC nodes on which user have *full "root" access*, running an operating system of user’s choice [28] [56].

Network resources are available to incorporate variety of experiments to test different hardware and software combinations required to execute experiments. The required resources for an experiment are specified in resource specifications, RSpec file in .xml or .rspec file format.

An experiment can be assembled if the availability of resources is known, available resources can be requested as per requirements, and resources can be allocated or promised to an experimenter [53]. Fig. 38 shows the interoperability of RSpecs in GENI. There are several issues about RSpecs as connectivity, topology, and expressiveness of the language to define resource specifications [53]. Rspec can identify every resource explicitly.



Type:	pc3000
Class:	pc
adminmfs_osid:	399
bios_waittime:	90
bootdisk_unit:	0
control_interface:	eth0
control_network:	0
default_imageid:	FBSD410+RHL90-STD
default_osid:	RHL-STD
delay_capacity:	2
delay_osid:	FBSD-STD
diskloadmfs_osid:	403
disksize:	146.00
disktype:	da
frequency:	3000
imageable:	1
jail_osid:	FBSD-STD
max_interfaces:	6
memory:	2048
power_delay:	60
processor:	64-bit Xeon
rebootable:	1
simnode_capacity:	0
trivlink_maxspeed:	10000000
virtnode_capacity:	100
virtnode_disksize:	2097152

Fig. 4.36: Detail description of a node type on Emulab Account [9].

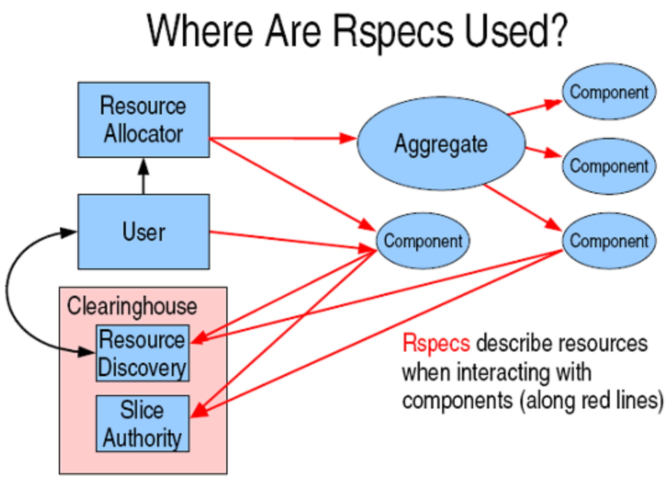
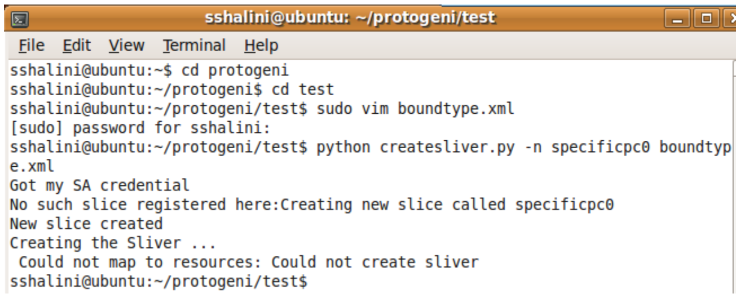


Fig. 4.37: Interoperability of RSpecs in GENI infrastructure [53]

Slice creation with Specific Type of Resources through RSpec

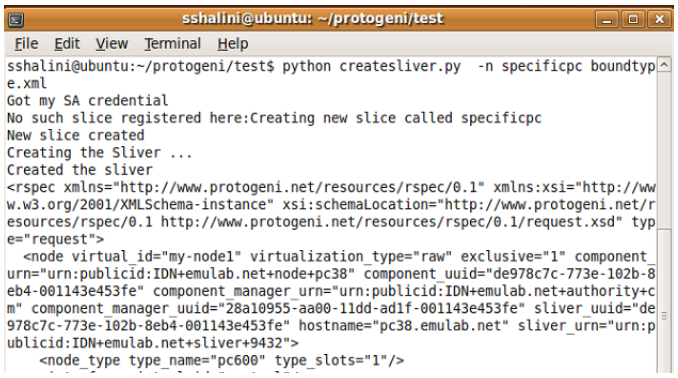
We tried to see how availability of resources can affect different network experiments. A file boundtype.xml was created from bound-type.rspec file [see Appendix] in test script package. It initially requested two PC2000 type PCs. As we can see from fig. 36, there was only one free node of PC2000 type, we could not create the slice. We again modified the RSpec details to request and included PC2400w with PC2000. There were 9 free PC2400w PCs but still, sliver could not be created. We tried to look for characteristics of PC2400w type PC and found its *virtnode_capacity* was defined as zero [9].


sshalini@ubuntu: ~/protogeni/test
File Edit View Terminal Help

```
sshalini@ubuntu:~$ cd protogeni
sshalini@ubuntu:~/protogeni$ cd test
sshalini@ubuntu:~/protogeni/test$ sudo vim boundtype.xml
[sudo] password for sshalini:
sshalini@ubuntu:~/protogeni/test$ python createsliver.py -n specificpc0 boundtyp
e.xml
Got my SA credential
No such slice registered here:Creating new slice called specificpc0
New slice created
Creating the Sliver ...
 Could not map to resources: Could not create sliver
sshalini@ubuntu:~/protogeni/test$
```

Fig. 4.38: Sliver creation attempt with PC type PC2400w

RSpec file was again modified to experiment with other types of nodes. 10 PC600 type nodes were free and node type virtnode_capacity was defined as 10, so we requested for two PC600 type PCs. Slice was created successfully.


sshalini@ubuntu: ~/protogeni/test
File Edit View Terminal Help

```
sshalini@ubuntu:~/protogeni/test$ python createsliver.py  -n specificpc boundtyp
e.xml
Got my SA credential
No such slice registered here:Creating new slice called specificpc
New slice created
Creating the Sliver ...
Created the sliver
<rspec xmlns="http://www.protogeni.net/resources/rspec/0.1" xmlns:xsi="http://ww
w.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.protogeni.net/r
esources/rspec/0.1 http://www.protogeni.net/resources/rspec/0.1/request.xsd" typ
e="request">
  <node virtual_id="my-node1" virtualization_type="raw" exclusive="1" component_
urn="urn:publicid:IDN+emulab.net+node+pc38" component_uuid="de978c7c-773e-102b-8
eb4-001143e453fe" component_manager_urn="urn:publicid:IDN+emulab.net+authority+c
m" component_manager_uuid="28a10955-aa00-11dd-ad1f-001143e453fe" sliver_uuid="de
978c7c-773e-102b-8eb4-001143e453fe" hostname="pc38.emulab.net" sliver_urn="urn:p
ublicid:IDN+emulab.net+sliver+9432">
    <node_type type_name="pc600" type_slots="1"/>
```

Fig. 4.39: Sliver creation with a specific type of PC600

Next section experiments are related to slice/ sliver creation and deletion to see how it affects availability of resources to other experiments. We tried to create specific RSpecs to see resource consumption with many slices, each with few resources and also stress test by requesting many resources in a single slice. Experiments show that one user can pose threat in execution of other experiments by holding resources in one or other way. One can make available resources not usable because of holding the necessary network resources in another experiment.

We are concerned about threats to resources for all ProtoGENI users. Sometimes a novice experimenter can create many slices to experience different basic experiments and may lost the track of slices created and deleted in process and may hold network resources. This unnoticed holding of resources may create shortage of resources. These resources on hold might be required for more important and preferable experiments to execute to see network behavior by mature experimenters. A beginner may not be capable to understand the severity of problem by holding few resources that could be resulted as bottleneck for another topology.

Resources can be acquired either by not specifying the particular type of node or a specific type of node can be specified in RSpec for a particular type of experiment. Though slices and slivers are short lived for few hours if not being renewed but still some basic experiments observing the effect of extending sliver time of existence through redeemticket.py may hold resources for a longer period which can block theses resources to be obtained by other ProtoGENI users. These experiments will help to understand the possible threats to availability of resources to other experimenters.

A sliver can be created with given slice name and RSpec.xml file name as following:

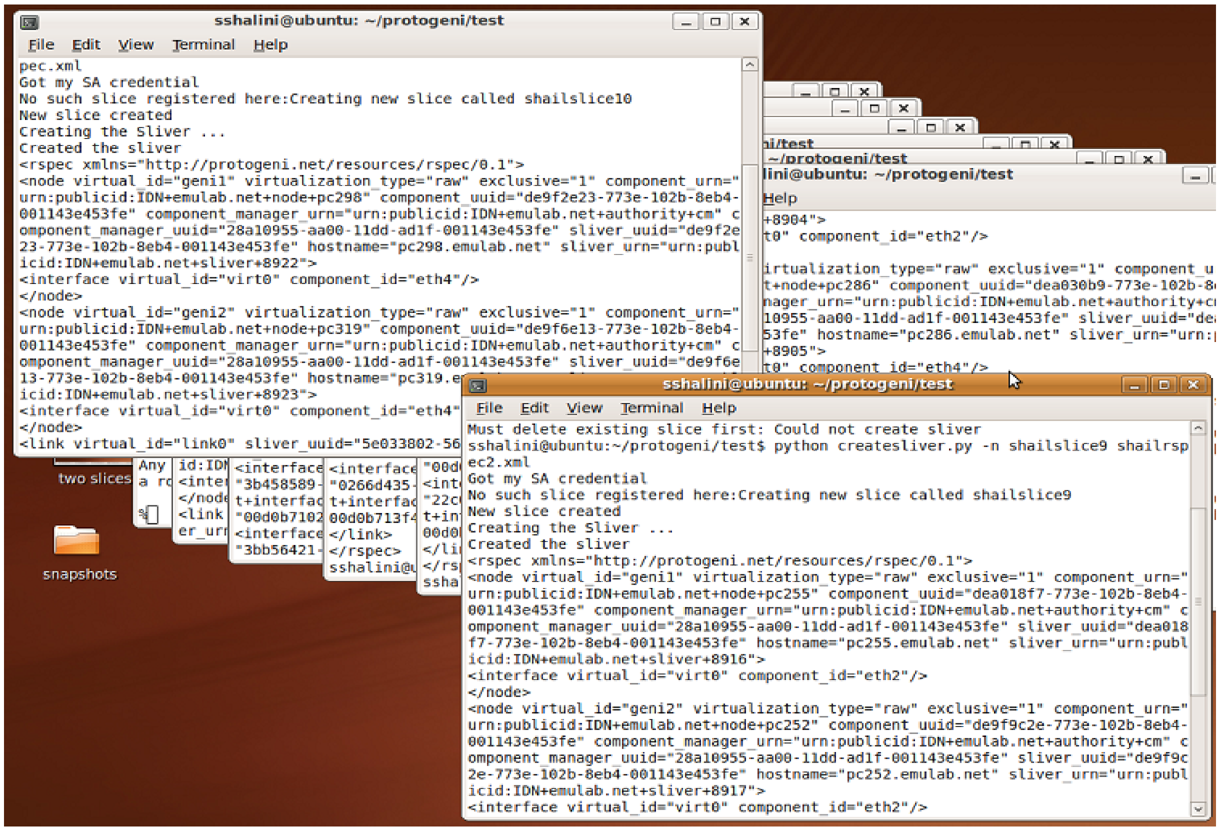
```
python  createsliver.py  –n myslice shailrspec.xml
```

It will create a sliver with required resources defined in shailrspec.xml file, and it will return the details of machines provided to user with node identification, which represents Emulab remote machines [9].

(a) Requesting few resources in too many slices/slivers

We observed Emulab account to keep tracking of available, consumed, and freed resources with creation and deletion of slices. Before starting of the experiment, 33 pcs were free on Emulab account so we decided to go up to16 Slices at first to see the resources consumption.

The sample RSpec in tutorial [28] was used for all slices which request only two PCs without stating any specific type. We tried to create one after another in separate terminals with a pattern of slice names like shailslice1, shailslice2 and so on. Initial 6 slices were created without any problem but 7th and 8th slices were stuck in creating slices and we aborted both of them, continuing creating next slices with same name pattern. Again, shailslice11 was not successfully created.


sshalini@ubuntu: ~/protogeni/test
File Edit View Terminal Help
two slices
snapshots

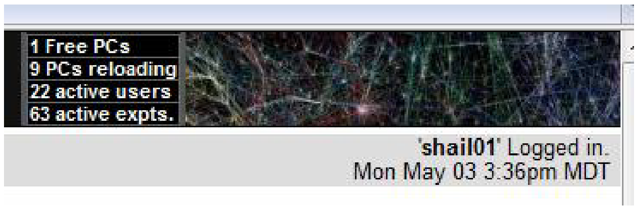
```
pec.xml
Got my SA credential
No such slice registered here:Creating new slice called shailslice10
New slice created
Creating the Sliver ...
Created the sliver
<rspec xmlns="http://protogeni.net/resources/rspec/0.1">
<node virtual_id="geni1" virtualization_type="raw" exclusive="1" component_urn="
urn:publicid:IDN+emulab.net+node+pc298" component_uuid="de9f2e23-773e-102b-8eb4-
001143e453fe" component_manager_urn="urn:publicid:IDN+emulab.net+authority+cm" c
omponent_manager_uuid="28a10955-aa00-11dd-ad1f-001143e453fe" sliver_uuid="de9f2e
23-773e-102b-8eb4-001143e453fe" hostname="pc298.emulab.net" sliver_urn="urn:publ
icid:IDN+emulab.net+sliver+8922">
<interface virtual_id="virt0" component_id="eth4"/>
</node>
<node virtual_id="geni2" virtualization_type="raw" exclusive="1" component_urn="
urn:publicid:IDN+emulab.net+node+pc319" component_uuid="de9f6e13-773e-102b-8eb4-
001143e453fe" component_manager_urn="urn:publicid:IDN+emulab.net+authority+cm" c
omponent_manager_uuid="28a10955-aa00-11dd-ad1f-001143e453fe" sliver_uuid="de9f6e
13-773e-102b-8eb4-001143e453fe" hostname="pc319.e
icid:IDN+emulab.net+sliver+8923">
<interface virtual_id="virt0" component_id="eth4"
</node>
<link virtual_id="link0" sliver_uuid="5e033802-56
```

```
+8904">
t0" component_id="eth2"/>

irtualization_type="raw" exclusive="1" component_u
t+node+pc286" component_uuid="dea030b9-773e-102b-8
nager_urn="urn:publicid:IDN+emulab.net+authority+c
10955-aa00-11dd-ad1f-001143e453fe" sliver_uuid="de
53fe" hostname="pc286.emulab.net" sliver_urn="urn:
+8905">
t0" component_id="eth4"/>
```

```
Must delete existing slice first: Could not create sliver
sshalini@ubuntu:~/protogeni/test$ python createsliver.py -n shailslice9 shailrsp
ec2.xml
Got my SA credential
No such slice registered here:Creating new slice called shailslice9
New slice created
Creating the Sliver ...
Created the sliver
<rspec xmlns="http://protogeni.net/resources/rspec/0.1">
<node virtual_id="geni1" virtualization_type="raw" exclusive="1" component_urn="
urn:publicid:IDN+emulab.net+node+pc255" component_uuid="dea018f7-773e-102b-8eb4-
001143e453fe" component_manager_urn="urn:publicid:IDN+emulab.net+authority+cm" c
omponent_manager_uuid="28a10955-aa00-11dd-ad1f-001143e453fe" sliver_uuid="dea018
f7-773e-102b-8eb4-001143e453fe" hostname="pc255.emulab.net" sliver_urn="urn:publ
icid:IDN+emulab.net+sliver+8916">
<interface virtual_id="virt0" component_id="eth2"/>
</node>
<node virtual_id="geni2" virtualization_type="raw" exclusive="1" component_urn="
urn:publicid:IDN+emulab.net+node+pc252" component_uuid="de9f9c2e-773e-102b-8eb4-
001143e453fe" component_manager_urn="urn:publicid:IDN+emulab.net+authority+cm" c
omponent_manager_uuid="28a10955-aa00-11dd-ad1f-001143e453fe" sliver_uuid="de9f9c
2e-773e-102b-8eb4-001143e453fe" hostname="pc252.emulab.net" sliver_urn="urn:publ
icid:IDN+emulab.net+sliver+8917">
<interface virtual_id="virt0" component_id="eth2"/>
```

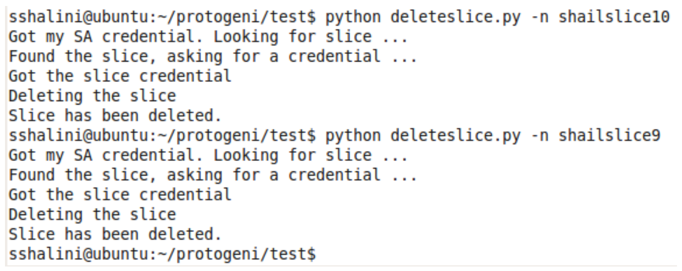
Fig.4.40: Creation of a series of slices from shailslice1 to shailslice16

shailslice13 and shailslice14 could not be created when tried in same terminal with previous slice but were successfully created when attempted in separate terminals. By the time of creation of shailslice15, there was only one free PC on Emulab records, and system didn't create shailslice15 as saying "could not map to resources, could not create the slice". Emulab resources were distributed among previous slices and left with one 1 free PC which could not help in creating a sliver with request of two PCs. Fig. 41 shows creation of a series of slices. Fig. 42, 43, and 44 show deletion of slices and resources being freed.



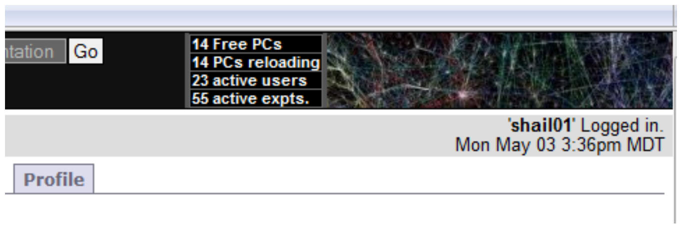
```
1 Free PCs
9 PCs reloading
22 active users
63 active expts.
'shail01' Logged in.
Mon May 03 3:36pm MDT
```

Fig. 4.41: Outage of ProtoGENI Resources through many sliver by single user



```
sshalini@ubuntu:~/protogeni/test$ python deleteslice.py -n shailslice10
Got my SA credential. Looking for slice ...
Found the slice, asking for a credential ...
Got the slice credential
Deleting the slice
Slice has been deleted.
sshalini@ubuntu:~/protogeni/test$ python deleteslice.py -n shailslice9
Got my SA credential. Looking for slice ...
Found the slice, asking for a credential ...
Got the slice credential
Deleting the slice
Slice has been deleted.
sshalini@ubuntu:~/protogeni/test$
```

Fig. 4.42-a: Deletion of slices to release the resources.



14 Free PCs
14 PCs reloading
23 active users
55 active expts.

'shail01' Logged in.
Mon May 03 3:36pm MDT

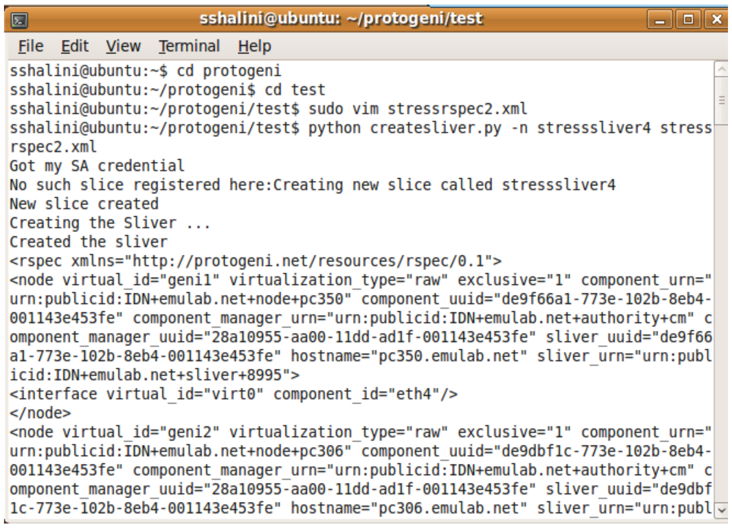
Fig. 4.42-b: More resources available with deletion of more slices.

Stress test

For stress test, we tried to request many resources at a time which again may reduce the possible number of experiments on Emulab resources. We modify the basic RSpec file to add more resources in one resource .xml file. We created stressrspec.xml [it is an expansion of basic RSpec with more resources in same format] with requirement of 6 PCs and 3 links at a time. Available resources were 33 PCs. Creation of two slices reduced the Emulab resources down to 21 free PCs.

RSpec file was modified to add more resources in one request. We created stresssrspec2.xml with the request of 14 Pcs and 7 links as shown in fig. 47. A slice was created with all 14 PCs and 7 links and Emulab resources reduced to 7 free PCs. Next attempt was to create a similar slice with 14 PCs and 7 links, it could not be created and stated: *"could not map to resources, could not create sliver"*.

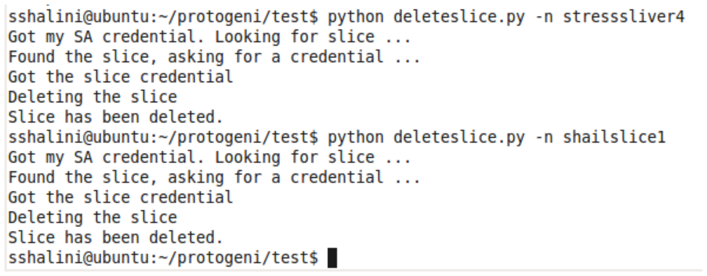
We can see that more resources requested at a time in a single slice reduces the chances of creation of other slivers. This poses a threat to availability of resources to experimenters.


sshalini@ubuntu: ~/protogeni/test
File Edit View Terminal Help

```
sshalini@ubuntu:~$ cd protogeni
sshalini@ubuntu:~/protogeni$ cd test
sshalini@ubuntu:~/protogeni/test$ sudo vim stressrspec2.xml
sshalini@ubuntu:~/protogeni/test$ python createsliver.py -n stresssliver4 stress
rspec2.xml
Got my SA credential
No such slice registered here:Creating new slice called stresssliver4
New slice created
Creating the Sliver ...
Created the sliver
<rspec xmlns="http://protogeni.net/resources/rspec/0.1">
<node virtual_id="geni1" virtualization_type="raw" exclusive="1" component_urn="
urn:publicid:IDN+emulab.net+node+pc350" component_uuid="de9f66a1-773e-102b-8eb4-
001143e453fe" component_manager_urn="urn:publicid:IDN+emulab.net+authority+cm" c
omponent_manager_uuid="28a10955-aa00-11dd-ad1f-001143e453fe" sliver_uuid="de9f66
a1-773e-102b-8eb4-001143e453fe" hostname="pc350.emulab.net" sliver_urn="urn:publ
icid:IDN+emulab.net+sliver+8995">
<interface virtual_id="virt0" component_id="eth4"/>
</node>
<node virtual_id="geni2" virtualization_type="raw" exclusive="1" component_urn="
urn:publicid:IDN+emulab.net+node+pc306" component_uuid="de9dbf1c-773e-102b-8eb4-
001143e453fe" component_manager_urn="urn:publicid:IDN+emulab.net+authority+cm" c
omponent_manager_uuid="28a10955-aa00-11dd-ad1f-001143e453fe" sliver_uuid="de9dbf
1c-773e-102b-8eb4-001143e453fe" hostname="pc306.emulab.net" sliver_urn="urn:publ
```

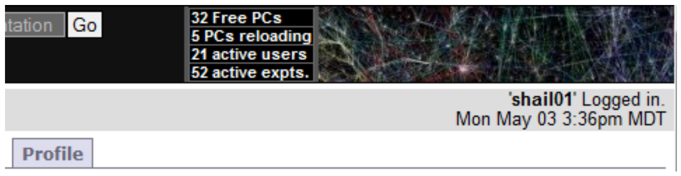
Fig. 4.43: Creation of stress slice with 14 PCs and 7 links.

We also observed the reverse activity of resources being freed with deletion of stress slices. This can be seen in fig. 4.44 and 4.45.



```
sshalini@ubuntu:~/protogeni/test$ python deleteslice.py -n stresssliver4
Got my SA credential. Looking for slice ...
Found the slice, asking for a credential ...
Got the slice credential
Deleting the slice
Slice has been deleted.
sshalini@ubuntu:~/protogeni/test$ python deleteslice.py -n shailslice1
Got my SA credential. Looking for slice ...
Found the slice, asking for a credential ...
Got the slice credential
Deleting the slice
Slice has been deleted.
sshalini@ubuntu:~/protogeni/test$ 
```

Fig. 4.44: Deletion of stress slices



32 Free PCs
5 PCs reloading
21 active users
52 active expts.

'shail01' Logged in.
Mon May 03 3:36pm MDT

Fig. 4.45: Free resources after deletion of stress slices.

The experiments show possible threats to availability of resources to GENI experimenters. It was taken in consideration that other experiments could create and delete slices when we were conducting these experiments, so exact number of resources available may be changed with the time. Fortunately, at the time of these experiments, there were few Emulab resources were free, and it was easier to observe the resource-outage with creation of fewer slices.

We also kept checking the control over acquired resources through sliver creations by doing login on those Emulab machines. Slices are generally short lived, so user has to renew his/her slice if the resources are required for a longer experiment. Resources were requested through RSpec.XML file, and RSpecs may be more complex and detailed for a particular experiment.

Existing ProtoGENI infrastructure is limited so experimenters need resources which may or may not be available because of involvement in other experiments. Availability is one important part of security issues and resources should be available to experimenters on equal basis.

Run-Time Interactions between ProtoGENI Components

Run-time network interactions can be a cause of network problems as different experiments interacting with different ProtoGENI components. Communication among the ProtoGENI nodes and also between ProtoGENI and outside network is important to explore further experiments. This section is focusing on experiments to test whether a ProtoGENI sliver can receive from (or send to) another slice, or outside network.

As we know that user can own resources in a sliver so he/ she can have control on those nodes, and can do the experiments. Now we are interested to see the communication capabilities between different slivers and between ProtoGENI sliver and outside network.

This can give us an idea about possible isolation or cooperation between ProtoGENI slivers and also between ProtoGENI sliver and outside network. This work can help to identifying the threats which could impact the accessibility of GENI.

We did experiments to find out the answers for following:

1. Whether a sliver can receive from (or send to) another slice?
2. Whether a sliver can receive from (or sent to) outside network?
3. Is communication possible between nodes from two component managers?
4. How communication between wireless nodes in different slivers can pose threats to network traffic?

We used simple approach to create different slivers and test the communication through simple client-server programs. We used our linux virtual machine (Ubuntu 9.0) as outside network node and verified the communication via client-server programs.

1. Communication between two ProtoGENI nodes as Client and Server:

Please see *Appendix for Client-Server program file Details*. We uploaded these files on remote machines. Server has to be initiated first.

At machine treated as Server:

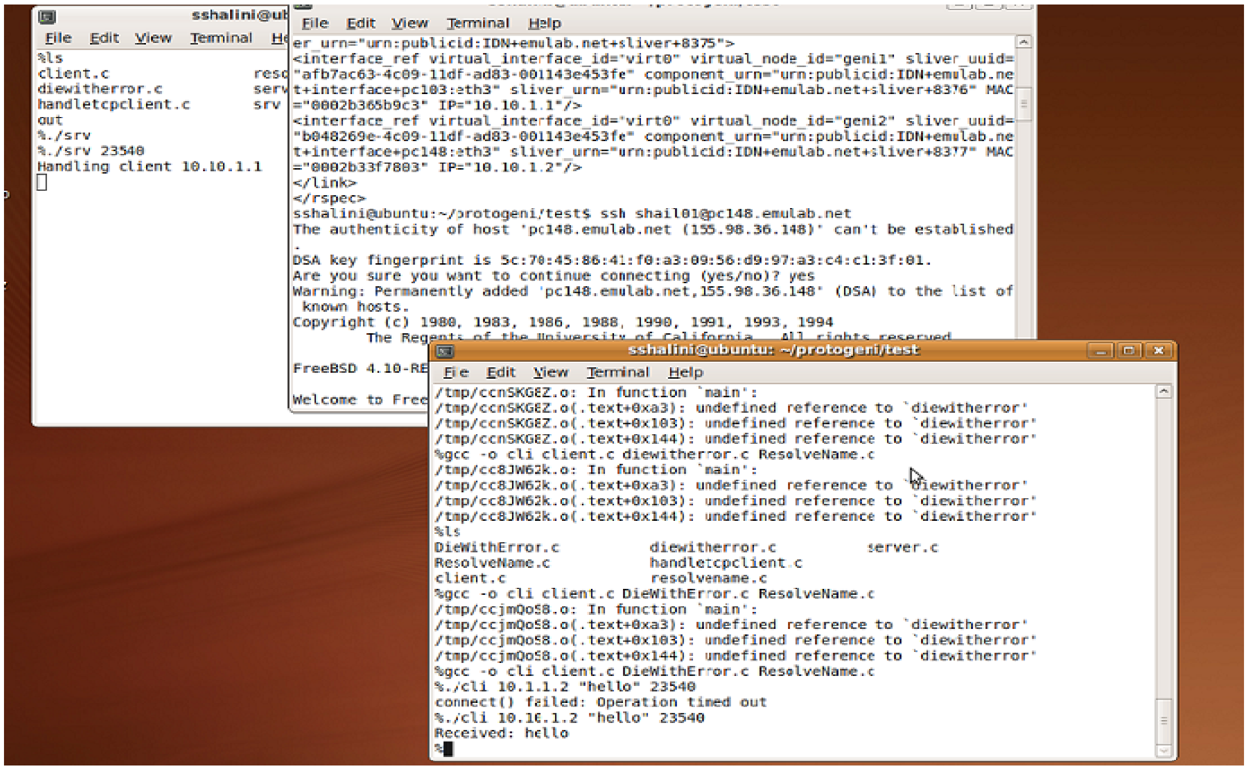
gcc -o srv server.c handletcpclient.c DieWithError.c

At machine treated as Client:

gcc -o cli client.c DieWithError.c ResolveName.c

After compilation, systems generate executable files to run client and server programs.

Communication between client and server: Communication can be seen as shown in fig. 49. We can run exit or logout on remote machines to close the connections like fig 50.

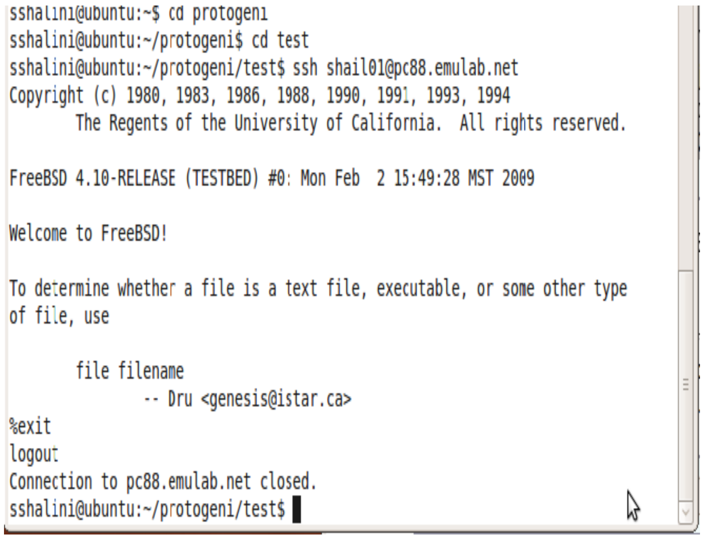


```
%ls
client.c                  reso
diewitherror.c            serv
handletcpclient.c         srv
out
%./srv
%./srv 23540
Handling client 10.10.1.1
```

```
er_urn="urn:publicid:IDN+emulab.net+sliver+8375">
<interface_ref virtual_interface_id="virt0" virtual_node_id="geni1" sliver_uuid=
"afb7ac63-4c09-11df-ad83-001143e453fe" component_urn="urn:publicid:IDN+emulab.ne
t+interface+pc103:eth3" sliver_urn="urn:publicid:IDN+emulab.net+sliver+8376" MAC
="0002b365b9c3" IP="10.10.1.1"/>
<interface_ref virtual_interface_id="virt0" virtual_node_id="geni2" sliver_uuid=
"b048269e-4c09-11df-ad83-001143e453fe" component_urn="urn:publicid:IDN+emulab.ne
t+interface+pc148:eth3" sliver_urn="urn:publicid:IDN+emulab.net+sliver+8377" MAC
="0002b33f7803" IP="10.10.1.2"/>
</link>
</rspec>
sshalini@ubuntu:~/protogeni/test$ ssh shail01@pc148.emulab.net
The authenticity of host 'pc148.emulab.net (155.98.36.148)' can't be established
.
DSA key fingerprint is 5c:70:45:86:41:f0:a3:09:56:d9:97:a3:c4:c1:3f:01.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'pc148.emulab.net,155.98.36.148' (DSA) to the list of
 known hosts.
Copyright (c) 1980, 1983, 1986, 1988, 1990, 1991, 1993, 1994
        The Regents of the University of California.  All rights reserved.

FreeBSD 4.10-RE

Welcome to Free
```

```
/tmp/ccnSKGEZ.o: In function `main':
/tmp/ccnSKGEZ.o(.text+0xa3): undefined reference to `diewitherror'
/tmp/ccnSKGEZ.o(.text+0x103): undefined reference to `diewitherror'
/tmp/ccnSKGEZ.o(.text+0x144): undefined reference to `diewitherror'
%gcc -o cli client.c diewitherror.c ResolveName.c
/tmp/cc8JW62k.o: In function `main':
/tmp/cc8JW62k.o(.text+0xa3): undefined reference to `diewitherror'
/tmp/cc8JW62k.o(.text+0x103): undefined reference to `diewitherror'
/tmp/cc8JW62k.o(.text+0x144): undefined reference to `diewitherror'
%ls
DieWithError.c          diewitherror.c          server.c
ResolveName.c           handletcpclient.c
client.c                resolvename.c
%gcc -o cli client.c DieWithError.c ResolveName.c
/tmp/ccjmQoS8.o: In function `main':
/tmp/ccjmQoS8.o(.text+0xa3): undefined reference to `diewitherror'
/tmp/ccjmQoS8.o(.text+0x103): undefined reference to `diewitherror'
/tmp/ccjmQoS8.o(.text+0x144): undefined reference to `diewitherror'
%gcc -o cli client.c DieWithError.c ResolveName.c
%./cli 10.1.1.2 "hello" 23540
connect() failed: Operation timed out
%./cli 10.10.1.2 "hello" 23540
Received: hello
%
```

Fig. 4.46: Communication between Client and Server machines



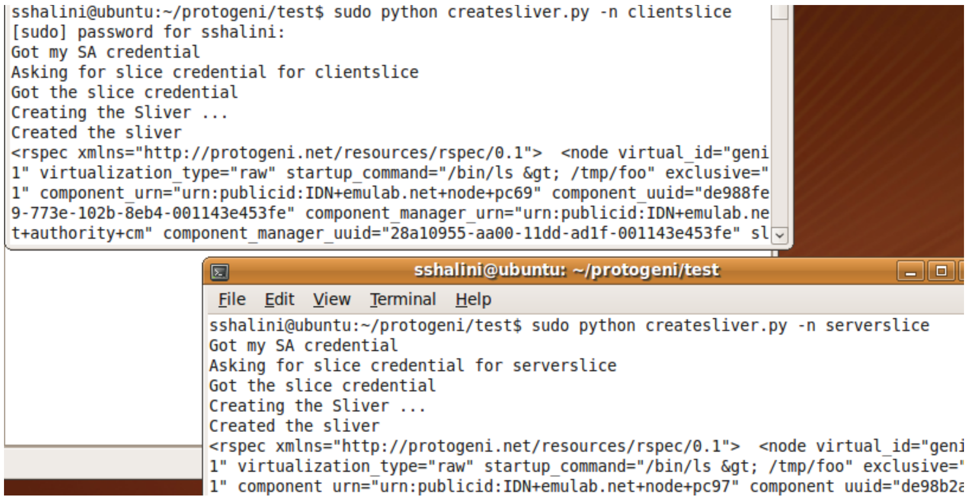
```
sshalini@ubuntu:~$ cd protogeni
sshalini@ubuntu:~/protogeni$ cd test
sshalini@ubuntu:~/protogeni/test$ ssh shail01@pc88.emulab.net
Copyright (c) 1980, 1983, 1986, 1988, 1990, 1991, 1993, 1994
        The Regents of the University of California.  All rights reserved.

FreeBSD 4.10-RELEASE (TESTBED) #0: Mon Feb  2 15:49:28 MST 2009

Welcome to FreeBSD!

To determine whether a file is a text file, executable, or some other type
of file, use

        file filename
                -- Dru <genesis@istar.ca>
%exit
logout
Connection to pc88.emulab.net closed.
sshalini@ubuntu:~/protogeni/test$
```

Fig. 4.47: Logout from remote Emulab machines

Now, resources are no longer in use so user should release the resources on priority basis. We can do it by running *deleteslice.py* command as per fig. 4.47.

2. *Communication between nodes of different slivers*

Two slices were created with no particular rspec file and system allowed one node to each sliver.



```
sshalini@ubuntu:~/protogeni/test$ sudo python createsliver.py -n clientslice
[sudo] password for sshalini:
Got my SA credential
Asking for slice credential for clientslice
Got the slice credential
Creating the Sliver ...
Created the sliver
<rspec xmlns="http://protogeni.net/resources/rspec/0.1">  <node virtual_id="geni
1" virtualization_type="raw" startup_command="/bin/ls &gt; /tmp/foo" exclusive="
1" component_urn="urn:publicid:IDN+emulab.net+node+pc69" component_uuid="de988fe
9-773e-102b-8eb4-001143e453fe" component_manager_urn="urn:publicid:IDN+emulab.ne
t+authority+cm" component_manager_uuid="28a10955-aa00-11dd-ad1f-001143e453fe" sl
```

sshalini@ubuntu: ~/protogeni/test
File Edit View Terminal Help

```
sshalini@ubuntu:~/protogeni/test$ sudo python createsliver.py -n serverslice
Got my SA credential
Asking for slice credential for serverslice
Got the slice credential
Creating the Sliver ...
Created the sliver
<rspec xmlns="http://protogeni.net/resources/rspec/0.1">  <node virtual_id="geni
1" virtualization_type="raw" startup_command="/bin/ls &gt; /tmp/foo" exclusive="
1" component_urn="urn:publicid:IDN+emulab.net+node+pc97" component_uuid="de98b2a
```

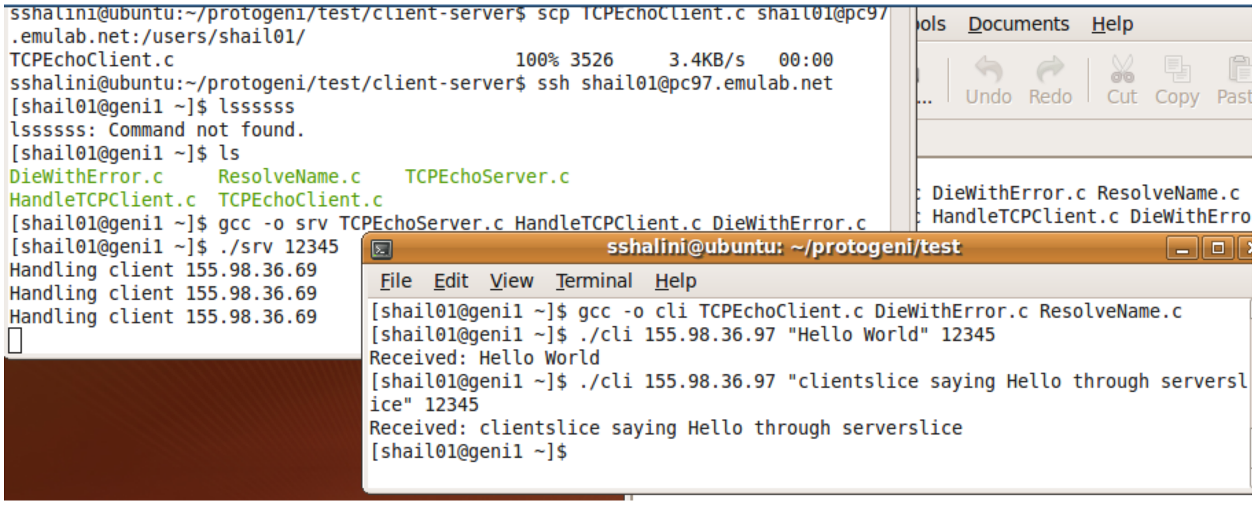
Fig. 4.48: Creation of two slivers for communication between slivers

PC69 were allotted to clientslice and PC97 was allotted to server slice.

We uploaded simple basic client server C program files on both nodes as we planned to observe both nodes as operating client as well as server. Files were uploaded one by one by following command with the required file name to upload:

sudo scp HandleTCPClient.c shail01@pc69.emulab.net:/users/shail01/

After this, roles were reversed for both slivers to check two ways operation between ProtoGENI slivers. PC97 was reset to act as client and PC69 was reset to act as client. Communication worked fine with reversed roles of sliver nodes.

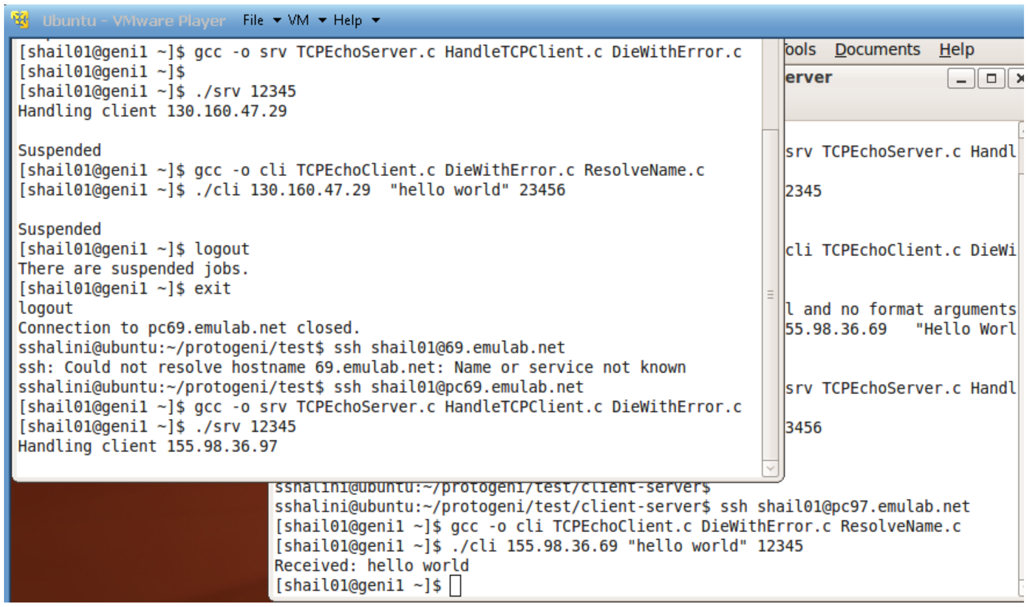

ols Documents Help
Undo Redo Cut Copy Past
sshalini@ubuntu: ~/protogeni/test
File Edit View Terminal Help

```
sshalini@ubuntu:~/protogeni/test/client-server$ scp TCPEchoClient.c shail01@pc97
.emulab.net:/users/shail01/
TCPEchoClient.c                              100% 3526     3.4KB/s   00:00
sshalini@ubuntu:~/protogeni/test/client-server$ ssh shail01@pc97.emulab.net
[shail01@geni1 ~]$ lssssss
lssssss: Command not found.
[shail01@geni1 ~]$ ls
DieWithError.c     ResolveName.c    TCPEchoServer.c
HandleTCPClient.c  TCPEchoClient.c
[shail01@geni1 ~]$ gcc -o srv TCPEchoServer.c HandleTCPClient.c DieWithError.c
[shail01@geni1 ~]$ ./srv 12345
Handling client 155.98.36.69
Handling client 155.98.36.69
Handling client 155.98.36.69
```

```
: DieWithError.c ResolveName.c
: HandleTCPClient.c DieWithErro
```

```
[shail01@geni1 ~]$ gcc -o cli TCPEchoClient.c DieWithError.c ResolveName.c
[shail01@geni1 ~]$ ./cli 155.98.36.97 "Hello World" 12345
Received: Hello World
[shail01@geni1 ~]$ ./cli 155.98.36.97 "clientslice saying Hello through serversl
ice" 12345
Received: clientslice saying Hello through serverslice
[shail01@geni1 ~]$
```

Fig. 4.49: Communication between two different sliver PCs



```
[shail01@geni1 ~]$ gcc -o srv TCPEchoServer.c HandleTCPClient.c DieWithError.c
[shail01@geni1 ~]$
[shail01@geni1 ~]$ ./srv 12345
Handling client 130.160.47.29

Suspended
[shail01@geni1 ~]$ gcc -o cli TCPEchoClient.c DieWithError.c ResolveName.c
[shail01@geni1 ~]$ ./cli 130.160.47.29  "hello world" 23456

Suspended
[shail01@geni1 ~]$ logout
There are suspended jobs.
[shail01@geni1 ~]$ exit
logout
Connection to pc69.emulab.net closed.
sshalini@ubuntu:~/protogeni/test$ ssh shail01@69.emulab.net
ssh: Could not resolve hostname 69.emulab.net: Name or service not known
sshalini@ubuntu:~/protogeni/test$ ssh shail01@pc69.emulab.net
[shail01@geni1 ~]$ gcc -o srv TCPEchoServer.c HandleTCPClient.c DieWithError.c
[shail01@geni1 ~]$ ./srv 12345
Handling client 155.98.36.97
```

```
srv TCPEchoServer.c Handl

2345

cli TCPEchoClient.c DieWi

l and no format arguments
55.98.36.69    "Hello Worl

srv TCPEchoServer.c Handl

3456
```

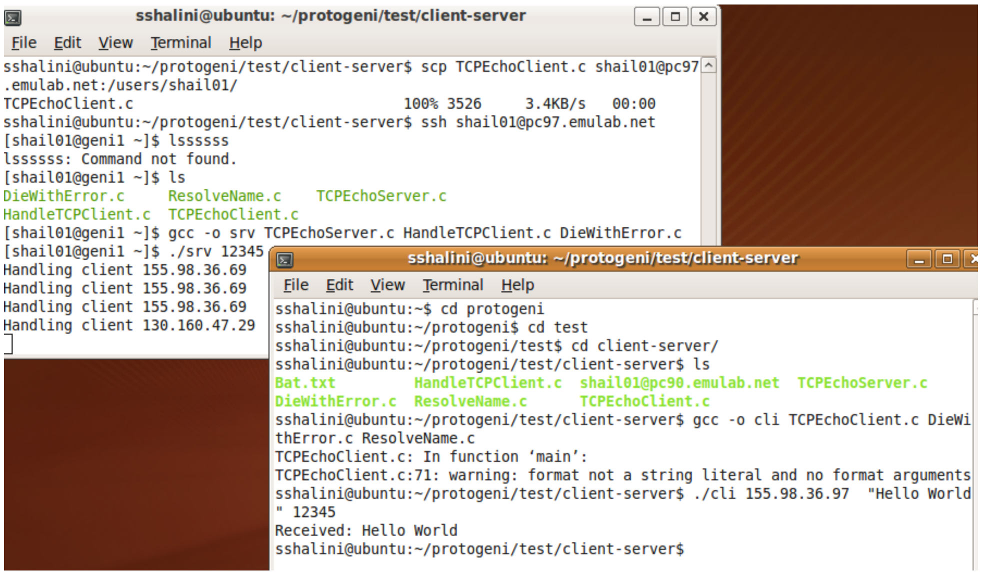
```
sshalini@ubuntu:~/protogeni/test/client-server$
sshalini@ubuntu:~/protogeni/test/client-server$ ssh shail01@pc97.emulab.net
[shail01@geni1 ~]$ gcc -o cli TCPEchoClient.c DieWithError.c ResolveName.c
[shail01@geni1 ~]$ ./cli 155.98.36.69 "hello world" 12345
Received: hello world
[shail01@geni1 ~]$
```

Fig. 4.50: verification of both slivers as client and server

These operations show that different slivers can send or receive from each other.

3. *Communication between a ProtoGENI sliver and outside network node*

We continued with same serverslice sliver with single node PC97 and opted for our own PC as outside network node to communicate with ProtoGENI sliver.


sshalini@ubuntu: ~/protogeni/test/client-server
File Edit View Terminal Help

```
sshalini@ubuntu:~/protogeni/test/client-server$ scp TCPEchoClient.c shail01@pc97
.emulab.net:/users/shail01/
TCPEchoClient.c                              100% 3526     3.4KB/s   00:00
sshalini@ubuntu:~/protogeni/test/client-server$ ssh shail01@pc97.emulab.net
[shail01@geni1 ~]$ lssssss
lssssss: Command not found.
[shail01@geni1 ~]$ ls
DieWithError.c      ResolveName.c    TCPEchoServer.c
HandleTCPClient.c  TCPEchoClient.c
[shail01@geni1 ~]$ gcc -o srv TCPEchoServer.c HandleTCPClient.c DieWithError.c
[shail01@geni1 ~]$ ./srv 12345
Handling client 155.98.36.69
Handling client 155.98.36.69
Handling client 155.98.36.69
Handling client 130.160.47.29
```

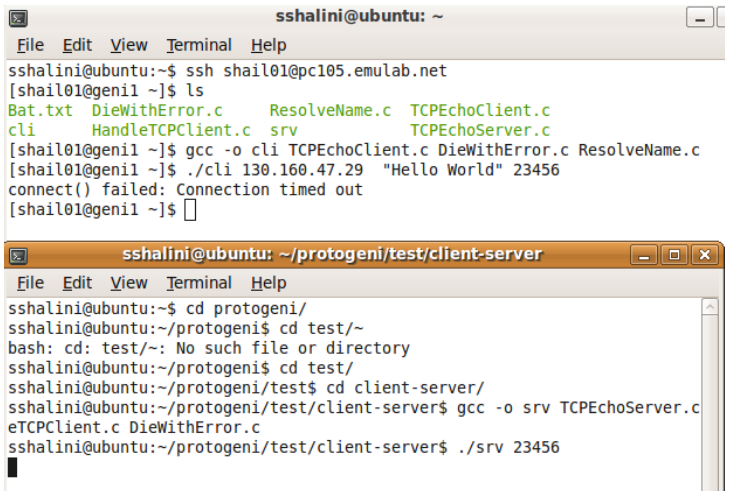
sshalini@ubuntu: ~/protogeni/test/client-server
File Edit View Terminal Help

```
sshalini@ubuntu:~$ cd protogeni
sshalini@ubuntu:~/protogeni$ cd test
sshalini@ubuntu:~/protogeni/test$ cd client-server/
sshalini@ubuntu:~/protogeni/test/client-server$ ls
Bat.txt          HandleTCPClient.c  shail01@pc90.emulab.net  TCPEchoServer.c
DieWithError.c   ResolveName.c      TCPEchoClient.c
sshalini@ubuntu:~/protogeni/test/client-server$ gcc -o cli TCPEchoClient.c DieWi
thError.c ResolveName.c
TCPEchoClient.c: In function ‘main’:
TCPEchoClient.c:71: warning: format not a string literal and no format arguments
sshalini@ubuntu:~/protogeni/test/client-server$ ./cli 155.98.36.97  "Hello World
" 12345
Received: Hello World
sshalini@ubuntu:~/protogeni/test/client-server$
```

Fig. 4.51: Communication of a ProtoGENI sliver with outside network

ProtoGENI sliver is operating as server and lab machine was used as outside network node. Communication is fine with this setting but connection could not be established when serverslice node was set as client. To diagnose a bit more we created the sliver again with shrspec1.xml and PC105 was allotted to experiment. Operational settings for both ProtoGENI node and outside network node were repeated to verify client and server operations but we got the same results. ProtoGENI node is working fine as server with an outside network client node but could not establish the connection when ProtoGENI node was reset as client and outside network node as server.

ProtoGENI sliver is working partially with outside network node. More work is required to find out the reason why it is not working as client with outside server node.


sshalini@ubuntu: ~
File Edit View Terminal Help

```
sshalini@ubuntu:~$ ssh shail01@pc105.emulab.net
[shail01@geni1 ~]$ ls
Bat.txt  DieWithError.c     ResolveName.c  TCPEchoClient.c
cli      HandleTCPClient.c  srv            TCPEchoServer.c
[shail01@geni1 ~]$ gcc -o cli TCPEchoClient.c DieWithError.c ResolveName.c
[shail01@geni1 ~]$ ./cli 130.160.47.29  "Hello World" 23456
connect() failed: Connection timed out
[shail01@geni1 ~]$
```

sshalini@ubuntu: ~/protogeni/test/client-server
File Edit View Terminal Help

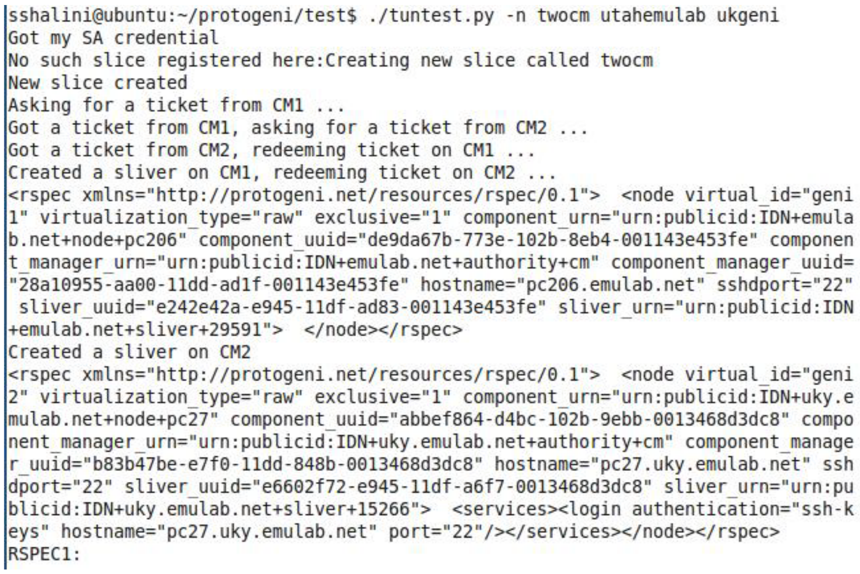
```
sshalini@ubuntu:~$ cd protogeni/
sshalini@ubuntu:~/protogeni$ cd test/~
bash: cd: test/~: No such file or directory
sshalini@ubuntu:~/protogeni$ cd test/
sshalini@ubuntu:~/protogeni/test$ cd client-server/
sshalini@ubuntu:~/protogeni/test/client-server$ gcc -o srv TCPEchoServer.c
eTCPClient.c DieWithError.c
sshalini@ubuntu:~/protogeni/test/client-server$ ./srv 23456
```

Fig. 4.52: Error in communication when ProtoGENI node acted as client

Though there is some clarification is required for not successful communication between a ProtoGENI node as client and outside network node as server, we expect that more experiments will be helpful to explore the reasons.

4. Communication between ProtoGENI nodes from two Component Managers

Sliver twocm was created with tuntest.py test script requesting two nodes from two different Component Managers (CM) Utah Emulab and Kentucky Emulab.



```
sshalini@ubuntu:~/protogeni/test$ ./tuntest.py -n twocm utahemulab ukgeni
Got my SA credential
No such slice registered here:Creating new slice called twocm
New slice created
Asking for a ticket from CM1 ...
Got a ticket from CM1, asking for a ticket from CM2 ...
Got a ticket from CM2, redeeming ticket on CM1 ...
Created a sliver on CM1, redeeming ticket on CM2 ...
<rspec xmlns="http://protogeni.net/resources/rspec/0.1">  <node virtual_id="geni
1" virtualization_type="raw" exclusive="1" component_urn="urn:publicid:IDN+emula
b.net+node+pc206" component_uuid="de9da67b-773e-102b-8eb4-001143e453fe" componen
t_manager_urn="urn:publicid:IDN+emulab.net+authority+cm" component_manager_uuid=
"28a10955-aa00-11dd-ad1f-001143e453fe" hostname="pc206.emulab.net" sshdport="22"
 sliver_uuid="e242e42a-e945-11df-ad83-001143e453fe" sliver_urn="urn:publicid:IDN
+emulab.net+sliver+29591">  </node></rspec>
Created a sliver on CM2
<rspec xmlns="http://protogeni.net/resources/rspec/0.1">  <node virtual_id="geni
2" virtualization_type="raw" exclusive="1" component_urn="urn:publicid:IDN+uky.e
mulab.net+node+pc27" component_uuid="abbef864-d4bc-102b-9ebb-0013468d3dc8" compo
nent_manager_urn="urn:publicid:IDN+uky.emulab.net+authority+cm" component_manage
r_uuid="b83b47be-e7f0-11dd-848b-0013468d3dc8" hostname="pc27.uky.emulab.net" ssh
dport="22" sliver_uuid="e6602f72-e945-11df-a6f7-0013468d3dc8" sliver_urn="urn:pu
blicid:IDN+uky.emulab.net+sliver+15266">  <services><login authentication="ssh-k
eys" hostname="pc27.uky.emulab.net" port="22"/></services></node></rspec>
RSPEC1:
```

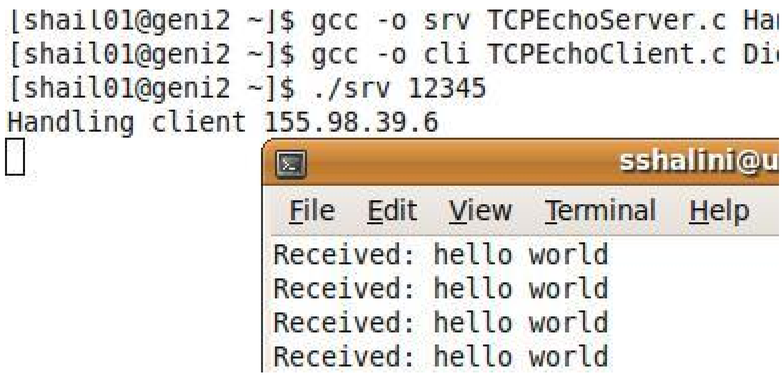
Fig. 4.53: Sliver creation requesting two nodes from two different CMs

Node geni1: Utah Emulab pc206.emulab.net;

Node geni2: Kentucky Emulab, pc27.uky.emulab.net

As before, we uploaded client server program files on remote nodes, and compiled the programs.

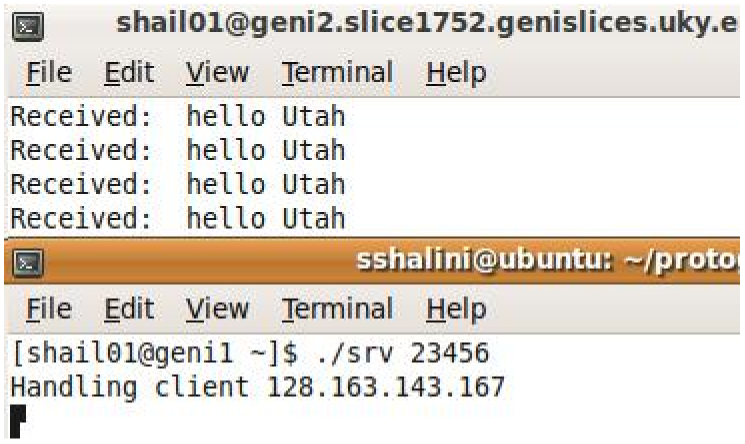
Communication was tried both ways: both nodes acting as server as well as client



```
[shail01@geni2 ~]$ gcc -o srv TCPEchoServer.c Ha
[shail01@geni2 ~]$ gcc -o cli TCPEchoClient.c Di
[shail01@geni2 ~]$ ./srv 12345
Handling client 155.98.39.6
```

```
Received: hello world
Received: hello world
Received: hello world
Received: hello world
```

Fig. 4.54: Kentucky Emulab ProtoGENI node as server



shail01@geni2.slice1752.genislices.uky.e
File Edit View Terminal Help

```
Received:  hello Utah
Received:  hello Utah
Received:  hello Utah
Received:  hello Utah
```

sshalini@ubuntu: ~/proto
File Edit View Terminal Help

```
[shail01@geni1 ~]$ ./srv 23456
Handling client 128.163.143.167
```

Fig. 4.55: Utah Emulab ProtoGENI node as server

5. Communication between Emulab wireless nodes

These experiments are to explore wireless traffic issues. Though these are not through slivers and initiated as Emulab experiments as more work is proposed on wireless nodes through ProtoGENI, this is initial work to interact with wireless nodes and to observe possible communication issues which will affect ProtoGENI experiments as well.

Two experiments were created to see communication between wireless nodes. One experiment had 4 wirelss nodes pcwf1, pcwf3, pcwf5, pcwf7 and second experiment was created with pcwf2 and pcwf4. Configuration of nodes can be verified to communicate via wireless channel through ifconfig command at remote nodes

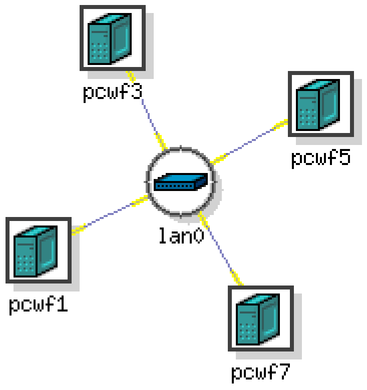
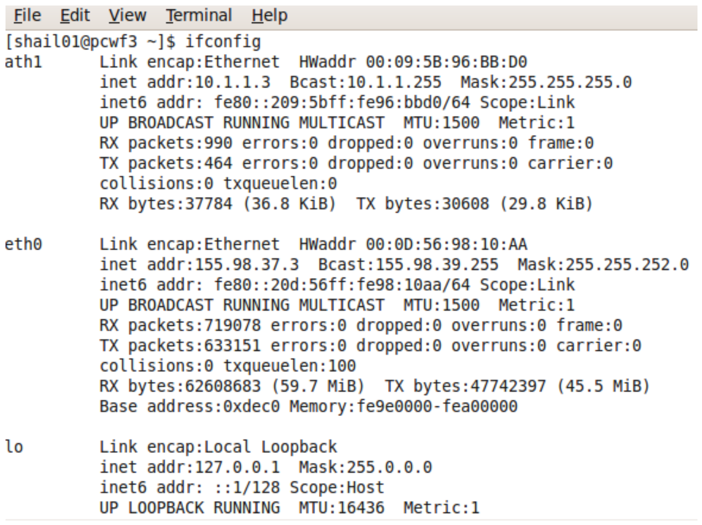


Fig. 4.56: Emulab topology for wireless nodes

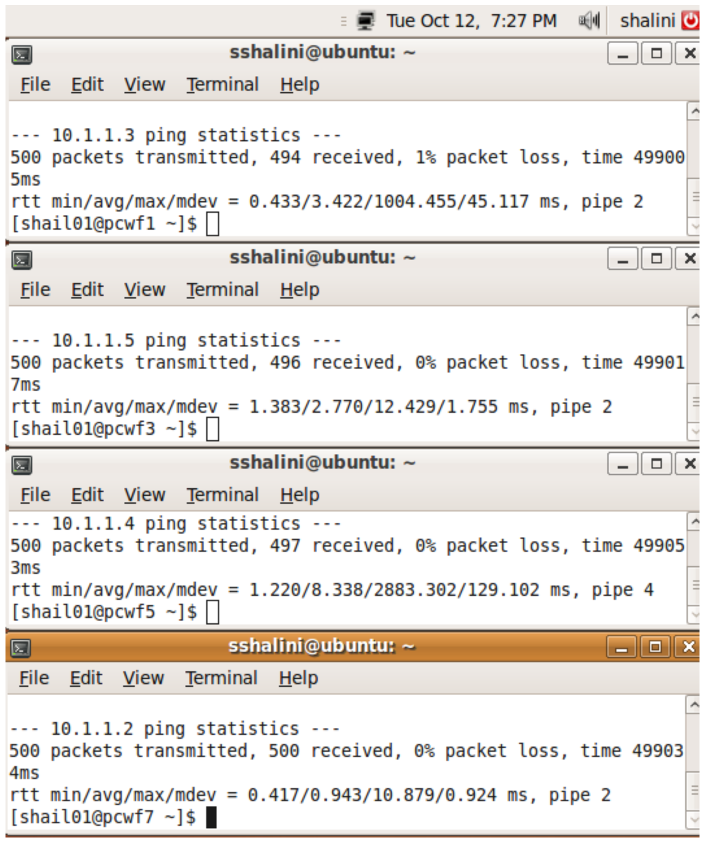


```
[shail01@pcwf3 ~]$ ifconfig
ath1      Link encap:Ethernet  HWaddr 00:09:5B:96:BB:D0
          inet addr:10.1.1.3  Bcast:10.1.1.255  Mask:255.255.255.0
          inet6 addr: fe80::209:5bff:fe96:bbd0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:990 errors:0 dropped:0 overruns:0 frame:0
          TX packets:464 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:37784 (36.8 KiB)  TX bytes:30608 (29.8 KiB)

eth0      Link encap:Ethernet  HWaddr 00:0D:56:98:10:AA
          inet addr:155.98.37.3  Bcast:155.98.39.255  Mask:255.255.252.0
          inet6 addr: fe80::20d:56ff:fe98:10aa/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:719078 errors:0 dropped:0 overruns:0 frame:0
          TX packets:633151 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:62608683 (59.7 MiB)  TX bytes:47742397 (45.5 MiB)
          Base address:0xdec0 Memory:fe9e0000-fea00000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
```

Fig. 4.57: Configuration details at remote node in wireless experiment

We tried to see the effect of pinging one node to other separately, pinging to every node together and effect of sending data in indefinite loop to see how these all affect traffic outputs and other communication process.

Table 2. in Appendix shows the pinging operations with different number of packets. Details show that max rtt (round trip time) is quite high with the increasing number of packets and especially when they were pinging all together. We observed that system applied different pipes to cope with the communication load but it increases the rtt significantly and loss of packets.



```
--- 10.1.1.3 ping statistics ---
500 packets transmitted, 494 received, 1% packet loss, time 49900
5ms
rtt min/avg/max/mdev = 0.433/3.422/1004.455/45.117 ms, pipe 2
[shail01@pcwf1 ~]$
```

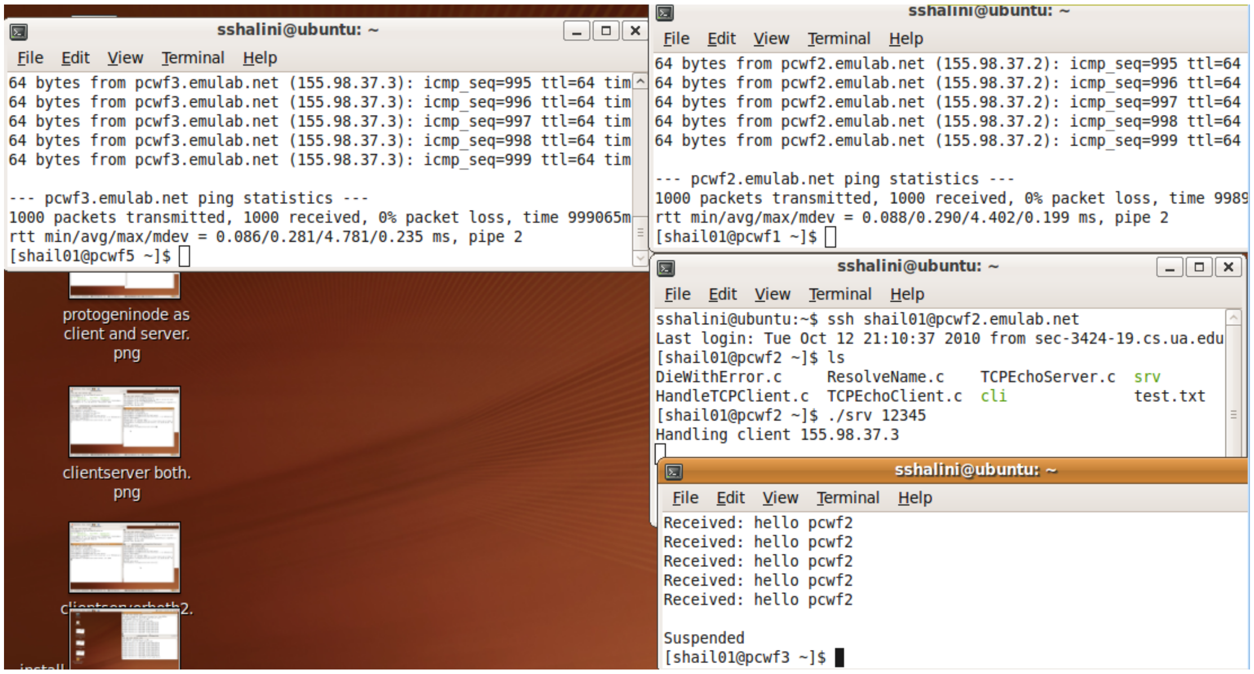
```
--- 10.1.1.5 ping statistics ---
500 packets transmitted, 496 received, 0% packet loss, time 49901
7ms
rtt min/avg/max/mdev = 1.383/2.770/12.429/1.755 ms, pipe 2
[shail01@pcwf3 ~]$
```

```
--- 10.1.1.4 ping statistics ---
500 packets transmitted, 497 received, 0% packet loss, time 49905
3ms
rtt min/avg/max/mdev = 1.220/8.338/2883.302/129.102 ms, pipe 4
[shail01@pcwf5 ~]$
```

```
--- 10.1.1.2 ping statistics ---
500 packets transmitted, 500 received, 0% packet loss, time 49903
4ms
rtt min/avg/max/mdev = 0.417/0.943/10.879/0.924 ms, pipe 2
[shail01@pcwf7 ~]$
```

Fig. 4.58: Pinging pcwf1-3-5-7-1, 500 pkts. (All at same time)

Client-server programs were tested while pinging and also tried when two nodes were communicating in indefinite loop and any other node tried to communicate with one of busy node. Cross communication between two experiments is shown as per fig. 4.59.



```
64 bytes from pcwf3.emulab.net (155.98.37.3): icmp_seq=995 ttl=64 tim
64 bytes from pcwf3.emulab.net (155.98.37.3): icmp_seq=996 ttl=64 tim
64 bytes from pcwf3.emulab.net (155.98.37.3): icmp_seq=997 ttl=64 tim
64 bytes from pcwf3.emulab.net (155.98.37.3): icmp_seq=998 ttl=64 tim
64 bytes from pcwf3.emulab.net (155.98.37.3): icmp_seq=999 ttl=64 tim

--- pcwf3.emulab.net ping statistics ---
1000 packets transmitted, 1000 received, 0% packet loss, time 999065m
rtt min/avg/max/mdev = 0.086/0.281/4.781/0.235 ms, pipe 2
[shail01@pcwf5 ~]$
```

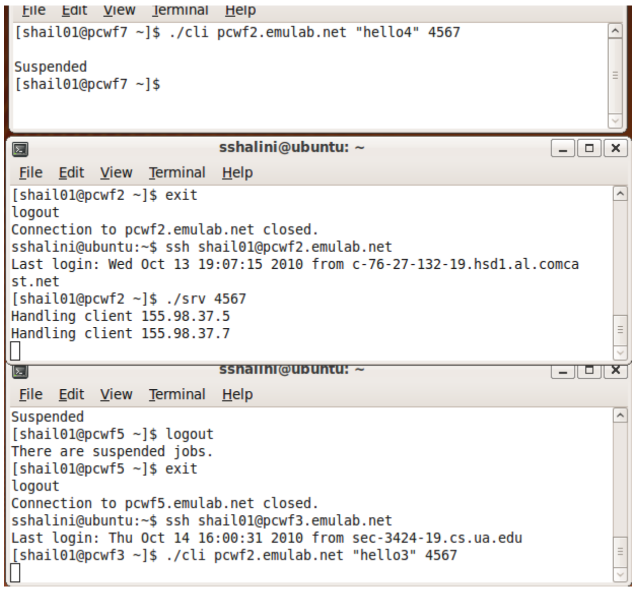
```
64 bytes from pcwf2.emulab.net (155.98.37.2): icmp_seq=995 ttl=64
64 bytes from pcwf2.emulab.net (155.98.37.2): icmp_seq=996 ttl=64
64 bytes from pcwf2.emulab.net (155.98.37.2): icmp_seq=997 ttl=64
64 bytes from pcwf2.emulab.net (155.98.37.2): icmp_seq=998 ttl=64
64 bytes from pcwf2.emulab.net (155.98.37.2): icmp_seq=999 ttl=64

--- pcwf2.emulab.net ping statistics ---
1000 packets transmitted, 1000 received, 0% packet loss, time 9989
rtt min/avg/max/mdev = 0.088/0.290/4.402/0.199 ms, pipe 2
[shail01@pcwf1 ~]$
```

```
sshalini@ubuntu:~$ ssh shail01@pcwf2.emulab.net
Last login: Tue Oct 12 21:10:37 2010 from sec-3424-19.cs.ua.edu
[shail01@pcwf2 ~]$ ls
DieWithError.c     ResolveName.c    TCPEchoServer.c  srv
HandleTCPClient.c  TCPEchoClient.c  cli              test.txt
[shail01@pcwf2 ~]$ ./srv 12345
Handling client 155.98.37.3
```

```
Received: hello pcwf2
Received: hello pcwf2
Received: hello pcwf2
Received: hello pcwf2
Received: hello pcwf2

Suspended
[shail01@pcwf3 ~]$
```

Fig. 4.59: Cross communication between two experiments with wireless nodes

Next, we tried to communicate with a busy node in an indefinite loop of client server communication.



```
[shail01@pcwf7 ~]$ ./cli pcwf2.emulab.net "hello4" 4567

Suspended
[shail01@pcwf7 ~]$
```

```
[shail01@pcwf2 ~]$ exit
logout
Connection to pcwf2.emulab.net closed.
sshalini@ubuntu:~$ ssh shail01@pcwf2.emulab.net
Last login: Wed Oct 13 19:07:15 2010 from c-76-27-132-19.hsd1.al.comca
st.net
[shail01@pcwf2 ~]$ ./srv 4567
Handling client 155.98.37.5
Handling client 155.98.37.7
```

```
Suspended
[shail01@pcwf5 ~]$ logout
There are suspended jobs.
[shail01@pcwf5 ~]$ exit
logout
Connection to pcwf5.emulab.net closed.
sshalini@ubuntu:~$ ssh shail01@pcwf3.emulab.net
Last login: Thu Oct 14 16:00:31 2010 from sec-3424-19.cs.ua.edu
[shail01@pcwf3 ~]$ ./cli pcwf2.emulab.net "hello3" 4567
```

Fig. 4.60: pcwf2 handling two clients

Experiments showed some different outputs like any third node was not able to communicate with the busy node but in certain cases when we close the client node, server showed handling of another client which was already been suspended. It was inconsistent as at other times it just denied the connection stating “connection was reset by peer”. More experiments will be helpful to diagnose theses issues.

Experiments show that different slivers can communicate and also can communicate to outside network. Wireless node communication experiments need more exploration but it can be seen that even pinging can affect the traffic behavior and communication can be affected if one user keeps busy another node in an indefinite loop.

SSH Security Issues in ProtoGENI

SSH is considered as a comparatively secure way to communicate remotely but there are many security issues associated with SSH. The protocol, default port settings are causes of security concerns. SSH attacks are among most repeated network security attacks. Port scanners are commonly used to find out open ports. Port configuration to a non-standard port, and disabling logins via SSH for the root account may help to increase the required work done in attacking the network. Experiments shows that port scanner like nmap, zenmap (GUI) can scan open ports and associated services. Default settings of SSH client and server can be vulnerable for attacks. Non-standard port for SSH and some other changes may help in protecting and reducing brute force attacks via SSH. Experiments show to follow certain steps to make a more secure SSH.

OpenSSH [68] is a SSH connectivity tool which encrypts all traffic details to efficiently reduce eavesdropping, connection hijacking, and other attacks.

OpenSSH-Client is installed with Ubuntu installation. OpenSSH-server was installed by following command: *sudo apt-get install openssh-server*

In this section, we will discuss the methods to activate these defense mechanisms. We executed theses experiment on Ubuntu 9, Open SSH [68] and setup created to use ProtoGENI slices [28] for experiments. By default Ubuntu installs only OpenSSH –client software on host so server configuration file could not be located on host. OpenSSH-server was installed as

Sudo apt-get install openssh-server

We also installed logwatch to see login details on host

Sudo apt-get install logwatch

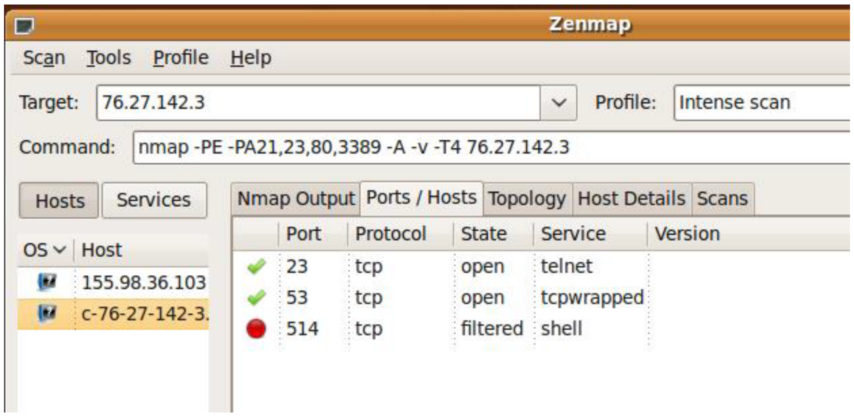
Port scanners can be helpful for network administration tasks like network inventory, managing service upgrade schedules, and monitoring host or service up time. Nmap, "Network Mapper", is one of the most popular port scanners. Nmap is free and open source, and it runs on all major operating systems [69]. Nmap has command line option as well as a very easy to use GUI zenmap as result viewer. Nmap [69] is quite popular among network security groups and hackers. We installed nmap and zenmap to scan host machine and also to scan acquired virtual Emulab machines.

To get port scanner

Sudo apt-get install nmap

Sudo apt-get install zenmap

Zenmap is GUI for nmap and has different types of scan. We used intense scan to get port details.

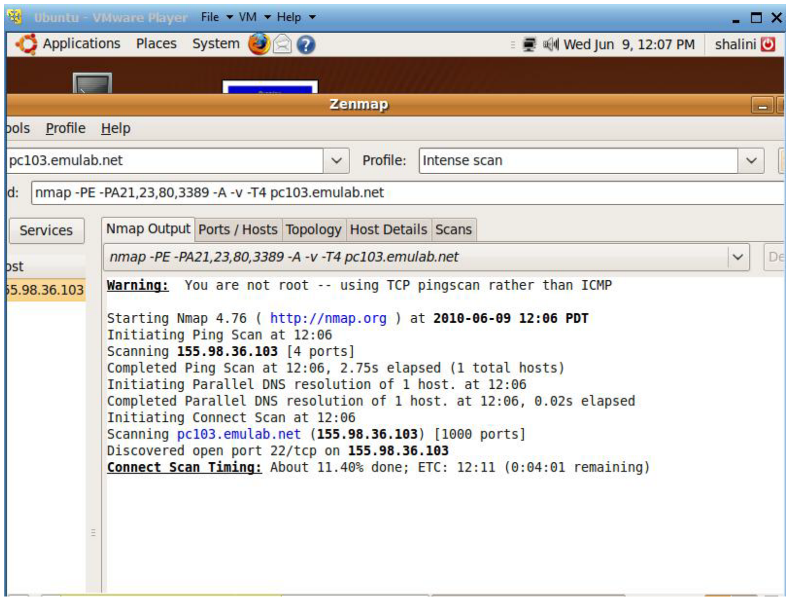


Target: 76.27.142.3
Profile: Intense scan
Command: nmap -PE -PA21,23,80,3389 -A -v -T4 76.27.142.3

OS	Host
	155.98.36.103
	c-76-27-142-3.

	Port	Protocol	State	Service	Version
✓	23	tcp	open	telnet	
✓	53	tcp	open	tcpwrapped	
●	514	tcp	filtered	shell	

Fig 4.61. Port Scanning details through Zenmap

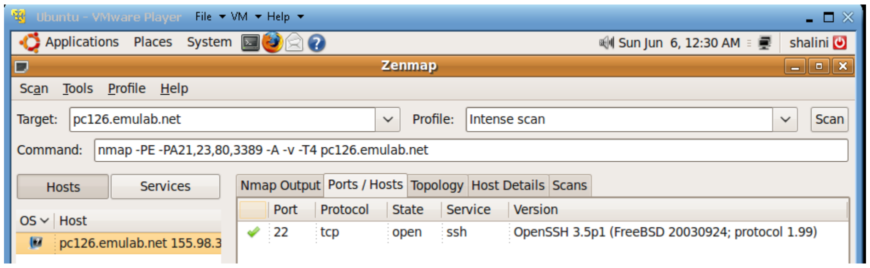


pc103.emulab.net
Profile: Intense scan
d: nmap -PE -PA21,23,80,3389 -A -v -T4 pc103.emulab.net

ost
55.98.36.103

```
nmap -PE -PA21,23,80,3389 -A -v -T4 pc103.emulab.net

Warning:  You are not root -- using TCP pingscan rather than ICMP

Starting Nmap 4.76 ( http://nmap.org ) at 2010-06-09 12:06 PDT
Initiating Ping Scan at 12:06
Scanning 155.98.36.103 [4 ports]
Completed Ping Scan at 12:06, 2.75s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 12:06
Completed Parallel DNS resolution of 1 host. at 12:06, 0.02s elapsed
Initiating Connect Scan at 12:06
Scanning pc103.emulab.net (155.98.36.103) [1000 ports]
Discovered open port 22/tcp on 155.98.36.103
Connect Scan Timing: About 11.40% done; ETC: 12:11 (0:04:01 remaining)
```

Fig 4.62: Nmap output after scanning an IP address



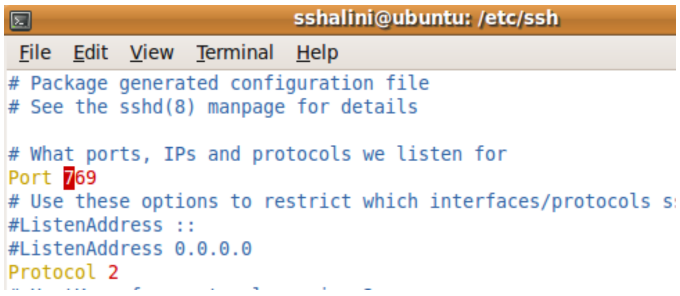
Target: pc126.emulab.net
Profile: Intense scan
Command: nmap -PE -PA21,23,80,3389 -A -v -T4 pc126.emulab.net

OS	Host
	pc126.emulab.net 155.98.3

	Port	Protocol	State	Service	Version
✓	22	tcp	open	ssh	OpenSSH 3.5p1 (FreeBSD 20030924; protocol 1.99)

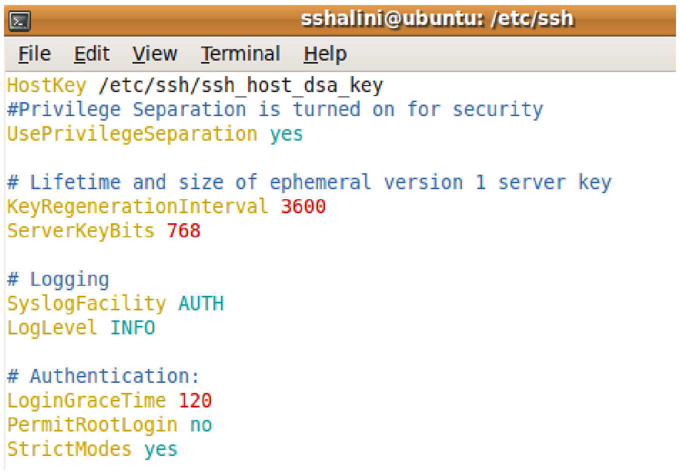
Fig 4.63: OS details of a machine through zenmap scan

User can go to /etc/ssh to find the sshd_config file to make changes. Port 22 is set as default for SSH services. This default setting is vulnerable to exploit through automated attacking tools. This port setting in sshd_config file can be changed to a suitable port number. SSH-server will listen on any unused port among 65,535 ports provided by the TCP protocols [61]. Current available port scanners are not capable to detect each and every detail of all 65,535 ports and usually developed to detect most common default settings and some other details so using a non-standard port number can be helpful to keep information secured which will reduce the attacking possibilities. File sshd_config can be modified as per fig 4.64.



```
# Package generated configuration file
# See the sshd(8) manpage for details

# What ports, IPs and protocols we listen for
Port 769
# Use these options to restrict which interfaces/protocols s
#ListenAddress ::
#ListenAddress 0.0.0.0
Protocol 2
```

Fig 4.64. Modification in sshd_config file for non-standard port for SSH



```
HostKey /etc/ssh/ssh_host_dsa_key
#Privilege Separation is turned on for security
UsePrivilegeSeparation yes

# Lifetime and size of ephemeral version 1 server key
KeyRegenerationInterval 3600
ServerKeyBits 768

# Logging
SyslogFacility AUTH
LogLevel INFO

# Authentication:
LoginGraceTime 120
PermitRootLogin no
StrictModes yes
```

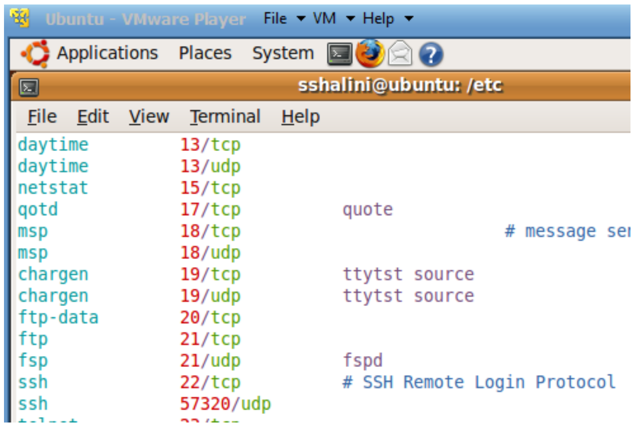
Fig 4.65: Restricting Root login via SSH

Nmap like scanning tools can scan around 1600 ports by default so other than default port settings it is a bit difficult to trace open non-standard ports. While it is not a preferred method to apply security but it works most of the time as experiments shows that there were no attacks after changing default SSH port settings from 22 to a non-standard port number while in same time period, over 100,000 attacks were reported on systems with default port settings [65]. The only additional work here seems to coordinate with all legitimate users to convey the right port settings for communication.

After setting the port 22 to a non standard port number, as shown in fig 70, settings should also be modified accordingly in /etc/services.

```
Home:~$ cd /etc
Home:/etc$ sudo vim services
```

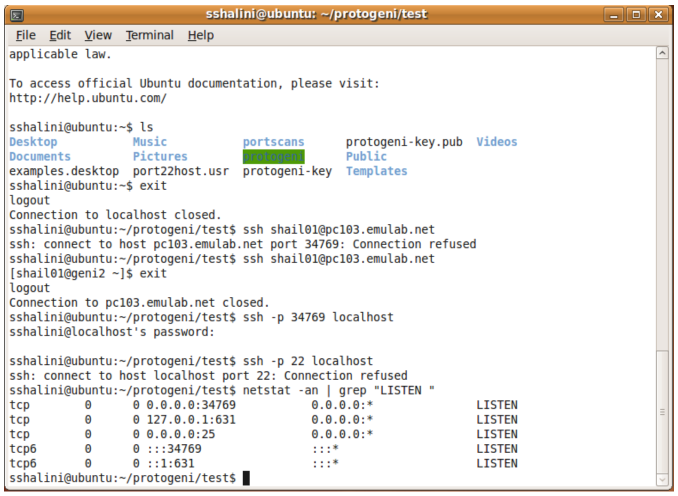
Here we changed only incoming port settings to new non-standard port for SSH connection because our remote Emulab machines have settings for default settings for SSH as port 22. We also modified PermitRootLogin as No in sshd_config file so now open port detection is difficult and there is also no permission to access SSH remotely, which is shown in fig 69. This exercise is done to make SSH a bit more secure and previous research supports and advocates customized settings to make it more difficult for attackers and not to be hacked by automated attacking scripts.



```
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
qotd            17/tcp          quote
msp             18/tcp                          # message se
msp             18/udp
chargen         19/tcp          ttytst source
chargen         19/udp          ttytst source
ftp-data        20/tcp
ftp             21/tcp
fsp             21/udp          fspd
ssh             22/tcp          # SSH Remote Login Protocol
ssh             57320/udp
```

Fig 4.66: Changing port no to non-standard port in services

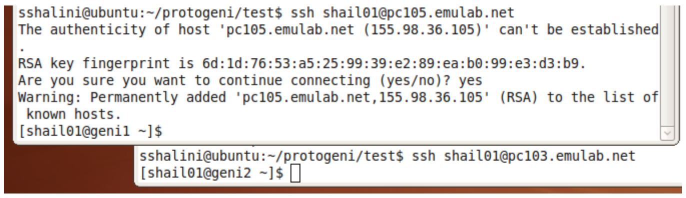
After completing all settings to work with new SSH non-standard port, we tried to login to Emulab machines. Before that we also verified that system is listening to that new port and could not be reached on default port 22 to access SSH as per fig 4.67 and fig 4.68.



```
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/

sshalini@ubuntu:~$ ls
Desktop           Music           portscans       protogeni-key.pub  Videos
Documents         Pictures        protogeni       Public
examples.desktop  port22host.usr  protogeni-key   Templates
sshalini@ubuntu:~$ exit
logout
Connection to localhost closed.
sshalini@ubuntu:~/protogeni/test$ ssh shail01@pc103.emulab.net
ssh: connect to host pc103.emulab.net port 34769: Connection refused
sshalini@ubuntu:~/protogeni/test$ ssh shail01@pc103.emulab.net
[shail01@geni2 ~]$ exit
logout
Connection to pc103.emulab.net closed.
sshalini@ubuntu:~/protogeni/test$ ssh -p 34769 localhost
sshalini@localhost's password:

sshalini@ubuntu:~/protogeni/test$ ssh -p 22 localhost
ssh: connect to host localhost port 22: Connection refused
sshalini@ubuntu:~/protogeni/test$ netstat -an | grep "LISTEN "
tcp        0      0 0.0.0.0:34769           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:25              0.0.0.0:*               LISTEN
tcp6       0      0 :::34769                :::*                    LISTEN
tcp6       0      0 ::1:631                 :::*                    LISTEN
sshalini@ubuntu:~/protogeni/test$
```

Fig 4.67: Verification of new port working as listening port

Now we verified the network operation to connect to Emulab machines in our sliver to check proper working of SSH with non-standard port.

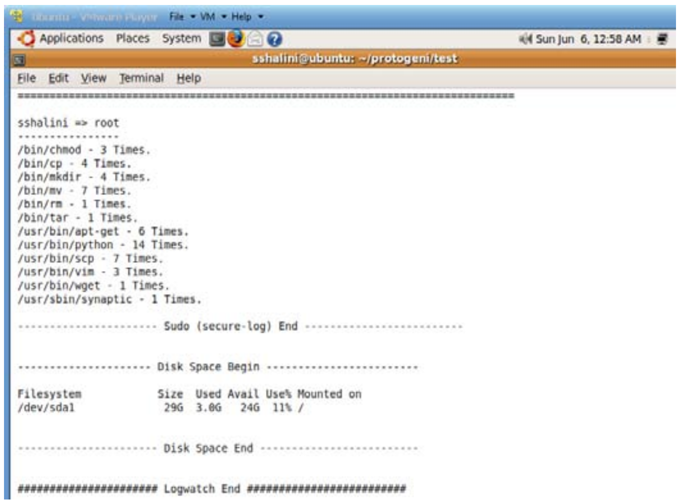


```
sshalini@ubuntu:~/protogeni/test$ ssh shail01@pc105.emulab.net
The authenticity of host 'pc105.emulab.net (155.98.36.105)' can't be established
.
RSA key fingerprint is 6d:1d:76:53:a5:25:99:39:e2:89:ea:b0:99:e3:d3:b9.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'pc105.emulab.net,155.98.36.105' (RSA) to the list of
 known hosts.
[shail01@geni1 ~]$
```

```
sshalini@ubuntu:~/protogeni/test$ ssh shail01@pc103.emulab.net
[shail01@geni2 ~]$
```

Fig 4.68: User login to Emulab machines with changed SSH port

System activities can be monitored through *logwatch* package as shown in fig 4.69.


Applications Places System
Sun Jun 6, 12:58 AM
sshalini@ubuntu: ~/protogeni/test
File Edit View Terminal Help

```
==================================================================================

sshalini => root
----------------
/bin/chmod - 3 Times.
/bin/cp - 4 Times.
/bin/mkdir - 4 Times.
/bin/mv - 7 Times.
/bin/rm - 1 Times.
/bin/tar - 1 Times.
/usr/bin/apt-get - 6 Times.
/usr/bin/python - 14 Times.
/usr/bin/scp - 7 Times.
/usr/bin/vim - 3 Times.
/usr/bin/wget - 1 Times.
/usr/sbin/synaptic - 1 Times.

---------------------- Sudo (secure-log) End ------------------------


--------------------- Disk Space Begin ------------------------

Filesystem            Size  Used Avail Use% Mounted on
/dev/sda1              29G  3.0G   24G  11% /


---------------------- Disk Space End ------------------------


###################### Logwatch End #########################
```

Fig 4.69: Logwatch details for observing activities

FUTURE WORK

These experiments deal with basic ProtoGENI functionality, and an effort to see the consumption and release pattern of Emulab resources with creation and deletion of slices and slivers. Future work is proposed to experiment with different combinations of wildcard allocation and specific kind of PCs in one RSpec. More study is needed to understand detailed description of Emulab node types and their characteristics to incorporate them in the RSpec to create desired network experiments.

Experimenters should know many details related to slices, slivers, and resource specifications and in-depth details about node type, their characteristics, and limitations of Emulab facility. This work focused on executing basic experiments with simple RSpecs to see the effect on availability of resources. It explains details of acquiring resources and holding of resources through many slices and also many resources in a single RSpec file to test availability of resources.

On verifying the vulnerability of wildcard allocation by RSpecs, we could not find an example to show because when many resources were free, it could take uncertain time to figure out the possible case when special kind of PCs were assigned to a sliver and then another experiment asking for that particular type of PC gets affected because of non availability of that resource. We tried to analyze possible situation through a network topology which shows that there might be a threat to availability of resources which would pose problems to other experimenters

More experiment will help to see actual effort and amount for an outsider to collect data and to perform attack. It would be helpful to create a setup as honey-pot and then analyzing outcomes before and after these SSH settings. We would like to work on possibility of defining a non-standard SSH port for all ProtoGENI nodes.

We analyzed SSH related work and tried to implement few modifications in default settings for SSH. Though we did not conduct any experiment to attack a system before and after making these adjustments but previous studies support a better security with these changes. Strong password practices, a bit attention on keeping physical machine protected, disabling the root login via SSH and changing default SSH port setting from port 22 to a non-standard port can help in reducing attacks via SSH.

CHAPTER 5

RESULTS, OBSERVATIONS, AND DISCUSSION

Security architecture for any system includes a number of security services like confidentiality, Integrity, availability, etc. [5] [45] [46] Roles, responsibilities, authorization, and authentication mechanisms in the system are also crucial for robust security. ProtoGENI experiments are conducted to observe the functionality and problems which can affect the ProtoGENI experiments.

5. Threats to availability of ProtoGENI resources
6. Problems in run-time interactions between ProtoGENI components
7. Security threats at host machine in ProtoGENI setup

Threats to Availability of ProtoGENI Resources:

1. *Attack through modifying test-common.py* script:

test-common.py test script is being used by all other scripts if attacker has access to the physical machine he can inject simple messages as print outputs without disturbing the actual process of script to fool users. If user sees a simple printing message in a convincing language, it can halt any further effort by user to request any resource or to do any other operation. Figure 4.7 shows that it can stop users to try any further and thus stopping the ProtoGENI experiments.

2. Non-Availability of ProtoGENI Resources:

1) ***Outage of Resources:*** Experiments show that ProtoGENI can face outage of resources for other experimenters if few experiments keep reserving resources without proper utilization. System did not give any warnings while all resources were being reserved by one user only.

Future work: This outage was for short time and after verification of problem, we released the resources, it may be subject of further exploration that after what time, system can identify this unusual resource reservation by one user.

Failing to map the resources as per RSpec : Many times, we couldn’t create a sliver as requested because it failed to map the requested resources. A particular type of pc was not free; a particular pc was down; node was free, available but couldn’t get for experiment. Third situation is certainly of concern as we couldn’t get the reason for not getting the resource. Characteristics of a certain pc type like *virtnode_capacity* also restricted the acquisition of a node in experiment.

One experiment couldn’t be activated as one fix node was required with other wireless nodes. It took almost 4 hr. while tried with several free and up nodes been shown on Emulab but system denied each time stating that fix node is not available. This problem indicates that even after system showing everything o.k. and good through sliverstatus.py, some other problems may halt the access to resources that can hurt the accessibility of resources.

Node status at Emulab-Observation: User can verify the node status before requesting it in a sliver. At Emulab, different type of pc-details pages have different color notations for representing the free, reserved, up, down status of nodes. Though it is specified on each page about color representation, it is confusing at times and user can misread the node status as changing with the pages. It will be very convenient if all pages may represent the node status with consistent color scheme and terminology.

Non-usability of ProtoGENI Resources:

Resources not usable: In some cases, resources were allotted to sliver but could not get hold on those nodes mainly because they were reported to be nodes with old OS image or old machines with not consistent performance. Sliver *sharednode* was created successfully and get two nodes virtually shared. Sliverstatus.py showed one node 'ready' and one as 'not ready' which did not come up even after hours so could not be used any further.

Suggestion: After defined period, system resources should be evaluated for their performance consistency and list of available resources and other related issues like OS images should be updated accordingly. Related error message or advising notes may also be helpful to choose the right resources to save time and effort, and moreover to remove the confusion clouds of "what went wrong"

Observations:

Slice/Sliver Creation: Many times, system got stuck somewhere in displaying the process of creation of sliver and never returned to normal. On screen, it seems that process is hung and only solution is to terminate it by closing the terminal. It is determined that even system doesn't show any progress or completion of slice/sliver creation, it completes it in the background. If status is checked for the same sliver in another terminal, it shows the existing sliver with its current status. This problem creates confusion and required to do this in different terminals. It consumed some time to understand the problem. Sometimes slivers in sequence couldn't be created in one terminal but when tried in a separate terminal, it worked fine.

Different variations to create slice and slivers are confusing and hard to differentiate for the benefits or problems associated with each. Resources can be acquired in a sliver without registering a slice first or also through getticket operation with or without registration of slice.

Currently, slice can handle only one sliver identical to slice name. Ticket operations were quite interesting as getticket provides a certain time to redeem the ticket but it expires within minutes. Even a deleted sliver became active when the ticket was redeemed (Fig. 20).

No ticket operation works once a slice/sliver is active so to renew the time for extending the experiment, user has to go through different path to renew the slice/sliver. We could not get the execution part for rleaseticket.py or unregisterslice.py.

Deleting the Slice/sliver and release of ProtoGENI Resources:

Resources are released with the deletion of sliver but sliver/slice is kept in system’s records until its expiration time at Clearinghouse. Test script showuser.py provides a list of slices for a particular user but include all such slices which are not expired but resources are released. If it includes only active slices holding the resources, it can help user to manage better.

Suggestion: New test script slicestatus.py stating the status of slice as empty/deleted or active

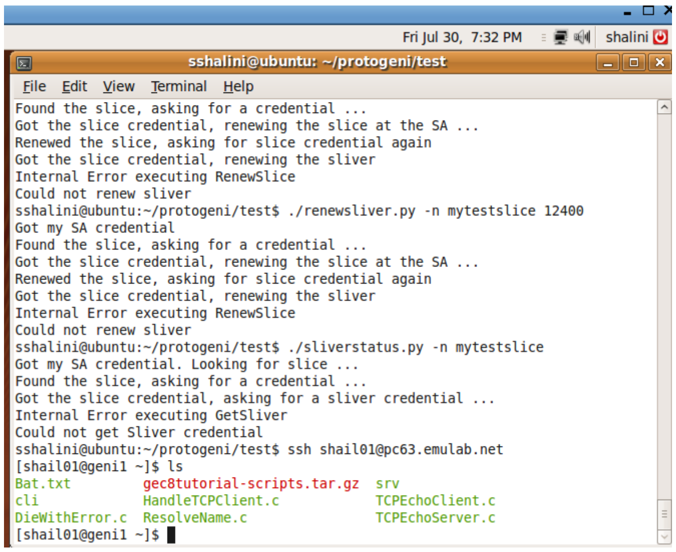
Manipulation Operations

UpdateSliver: This test script, updatesliver.py is supposed to modify the sliver resources without deleting the sliver. In trial, the sliver is being updated in records and returns the details on screen with information to redeem before a particular time but sliver status was showing the same details as before applying the updatesliver.py. The updatesliver.py command was repeated with a different RSpec file for a single node but it could not create geniobject, and sliver could not be started or revived further. Updating a sliver is a bit complex as it involves quick redemption of new ticket. A detailed description of outputs is included in Appendix.

Alternate temporary solution may be to create a new sliver other than updating the existing sliver.

System bugs: Problem observed in execution of all test scripts

Sometimes bugs interfere and halt the system. During trial of renewal a slice, test script renewsliver.py did not work. No other test scripts could be executed because of a bug. The problem was sent to ProtoGENI people. This bug halted the basic functionality for more than 12 hours.



Fri Jul 30, 7:32 PM
shalini
sshalini@ubuntu: ~/protogeni/test
File Edit View Terminal Help

```
Found the slice, asking for a credential ...
Got the slice credential, renewing the slice at the SA ...
Renewed the slice, asking for slice credential again
Got the slice credential, renewing the sliver
Internal Error executing RenewSlice
Could not renew sliver
sshalini@ubuntu:~/protogeni/test$ ./renewsliver.py -n mytestslice 12400
Got my SA credential
Found the slice, asking for a credential ...
Got the slice credential, renewing the slice at the SA ...
Renewed the slice, asking for slice credential again
Got the slice credential, renewing the sliver
Internal Error executing RenewSlice
Could not renew sliver
sshalini@ubuntu:~/protogeni/test$ ./sliverstatus.py -n mytestslice
Got my SA credential. Looking for slice ...
Found the slice, asking for a credential ...
Got the slice credential, asking for a sliver credential ...
Internal Error executing GetSliver
Could not get Sliver credential
sshalini@ubuntu:~/protogeni/test$ ssh shail01@pc63.emulab.net
[shail01@geni1 ~]$ ls
Bat.txt         gec8tutorial-scripts.tar.gz  srv
cli             HandleTCPClient.c            TCPEchoClient.c
DieWithError.c  ResolveName.c                TCPEchoServer.c
[shail01@geni1 ~]$
```

Fig. 5.1: System bug created problem in executing test scripts

Here observation was that the acquired node was active and good to login but test scripts were not working. Other thing is that though bug stopped the new operations on ProtoGENI, users could utilize the resources which were already acquired and up for experiments.

As we are concerned about the threats to ProtoGENI resources, experiments show that though system is basically protected with encryption, authorization, and authentication but system is not consistent in certain functions and prone to bugs which can halt the whole availability and accessibility of ProtoGENI resources. System is also prone to be attacked if host system is compromised and a corrupt test-common.py can affect all other test scripts and so the functionality of ProtoGENI.

Availability and accessibility of resources is very important to do any experiments further to see the network behavior and to identify the ways to breach the system which can be used to improve the overall ProtoGENI functionality.

Emulab resources are allocated to a single user at a time. If nodes are being used by some experimenters, it may hinder the possibility of other experiments to run because of unavailability of resources or because of non-usability of resources for different reasons as described above.

Sometimes a novice experimenter can create many slices to experience different basic experiments and may lost the track of slices created and deleted in process and may hold network resources. This unnoticed holding of resources may create shortage of resources. These resources on hold might be required for more important and preferable experiments to execute to see network behavior by mature experimenters. A beginner may not be capable to understand the severity of problem by holding few resources that could be resulted as bottleneck for another topology. Though slices and slivers are short lived for few hours if not being renewed but still some basic experiments observing the effect of extending sliver time of existence through redeemticket.py may hold resources for a longer period which can block theses resources to be obtained by other ProtoGENI users. This poses a threat to resources available to experimenters.

Vulnerability of Wildcard Specification in RSpecs

Emulab has different types of resources, and experimenters can define their required type of resources through specifying the details in RSpecs. Sometimes, it may be hard to find the required specific resources because of other running experiments. It is also complex to know each and every detail about the characteristics of a node type to make sure the adequate choice for experiments. Especially, novice experimenters may not understand complexity of details, so it is a general practice to use wildcard allocation of available resources.

It is simple and repeatable process to get resources of Emulab based on the best availability. In certain cases, system may assign some special type of PCs which are very few in numbers, and because of holding of resources, they may be unavailable for some time depending on experimenter acquired those PCs. This can cause a bottleneck for other experimenters who may need that special kind of PC in their topology. Therefore, many other available resources would not help as sliver cannot be created without availability of all asked resources in RSpecs.

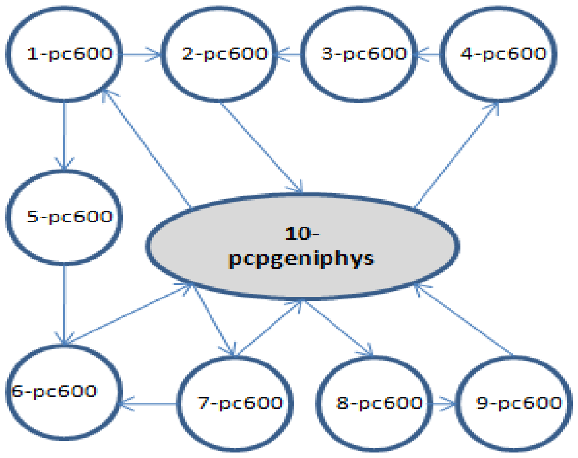


Fig. 5.2: A network topology which may suffer due to wildcard allocation of resources.

As shown in fig. 4.71, a network topology can suffer by unavailability of resources because of wildcard allocation through RSpecs. In this example, if 10- pcpgeniphys has allotted to some other experiment through wildcard, this topology will be very restricted to perform the desired operations or interconnections among all nodes of topology.

For example, if 10-pcpgeniphys is not available, 6-pc600 will be almost isolated with all other nodes. 2,3,4 pc600 and 1,5,6 pc600 can have some usability in this topology. With the availability of 10-pcpgeniphys, the network topology may have much more interaction and functionality possible in the experiments.

Discussion on Results and Observations:

The experiments show possible threats to availability of resources to GENI experimenters. A novice experimenter may be more prone to lost track of created slices and also not to know all details which can be an issue in designing and executing the experiment. Sometimes, slices could not be created as quickly as otherwise and updating of free resources may not be exact.

System may or may not update its updating on aborted sliver very quickly which would also affect updating of available resources. It is also observer that other experiments could create and delete slices when we were conducting these experiments, so exact number of resources available may be changed with the time. Fortunately, at the time of these experiments, there were few Emulab resources were free, and it was easier to observe the resource-outage with creation of fewer slices.

We also kept checking the control over acquired resources through sliver creations by doing login on those Emulab machines. Slices are generally short lived, so user has to renew his/her slice if the resources are required for a longer experiment. Resources were requested through RSpec.XML file, and RSpecs may be more complex and detailed for a particular experiment but we applied simple RSpec .xml files.

Run-time Interactions between ProtoGENI Components:

Communication between nodes of same sliver: it was good in both ways as client as well as server

Communication between nodes of two slivers: good in both ways

Communication between nodes of two Component Managers in two slivers: good in both ways

Communication with outside network and ProtoGENI sliver: good when ProtoGENI sliver node acts as server but couldn’t establish connection when tried in reverse.

Though there is some clarification is required for not successful communication between a ProtoGENI node as client and outside network node as server, we expect that more experiments will be helpful to explore the reasons.

Wireless Communication in Emulab: Traffic load can force system resources to utilize its resources at best in concurrency which may lead to significant delays in some processes and loss of packets. It can also affect the communication efforts by other network nodes. Again there are some unsolved issues and need more experiments. ProtoGENI uses the same resources in same manner but in sliver. Further work is supposed to repeat settings in ProtoGENI sliver settings.

Observation: All wireless nodes with required specification were busy in other experiments and we had to wait for 6 days then a request was sent to Emulab people and they provided the required type of nodes by reserving them.

Attack via stealing user credentials

Passphrase: Once a sliver is created and resources are acquired, passphrase can be deleted from the host machine through forgetpassphrase.py. User can login to remote nodes and can do further actions which leaves the host machine a bit more secured as there is no passphrase to steal.

Attack via SSH

Experiments explore the problems related to SSH (Secure Shall), possible ways to attack via SSH and methods to restrict attackers to attack via SSH like changing the default SSH port no and restricting other users as root user.

More experiments are required to see actual effort and amount for an outsider to collect data and to perform attack. It would be helpful to create a setup as honey-pot and then analyzing outcomes before and after these SSH settings.

We would like to work on possibility of defining a non-standard SSH port for all ProtoGENI nodes. Further experiments can explore the better and more secure ways to get and utilize ProtoGENI resources.

SUMMARY

All components of ProtoGENI should be in function with their intended roles to get authentic results to analyze for developing a more secure and accountable future Internet. As GENI/ProtoGENI enjoys deep programmability, it may be a cause of much disturbance once any part is broken intentionally or accidently. Accessibility and availability of resources is critical to experimenters as initial thing because if there are no resources available for experiments or resources can be compromised by exploiting different vulnerabilities in GENI/ ProtoGENI system, it will either disturb the basic facility for experiments or it will add questions to the results of experiment’s results. If any abnormal processing is unnoticed, it may affect the overall output. Experiments show that though system is basically protected with encryption, authorization, and authentication but system is not consistent in certain functions and prone to bugs which can halt the whole availability and accessibility of ProtoGENI resources. System is also can be attacked if host system is compromised and a corrupt test-common.py can affect all other test scripts and so the functionality of ProtoGENI. There may be outage of resources because of non-availability or non-usability of resources. Certain resources are not consistent and it takes much time and effort to figure out what went wrong. Few functions also did not perform as expected, and sometimes resources were being shown free but could not be acquired. Slice/Sliver creation functionality and associated ticket operations are a bit complex to understand because of duplication of same function in different ways.

Renewing of slivers and deletion of slivers are easy operations but their recording and updating on clearinghouse can create confusion about releasing the resources or using a sliver name in particular.

System bug halted the whole ProtoGENI system for many hours which may be real problem for some experiments as some resources, already in use, can be reset, and user can be asked to discontinue the experiment. SSH security issues are known and can affect the ProtoGENI functioning after misusing the information available through default SSH settings and Port scanners. Some modifications at host machine can enhance the SSH security at host machine but a tough password and restricted physical access to host machine may be helpful to enhance overall ProtoGENI security.

REFERENCES

[1] Howard F. Lipson, PhD , “Tracking and Tracing Cyber-Attacks: Technical Challenges and Global Policy Issues, CERT® Coordination Center, November 2002, SPECIAL REPORT

[2] http://www.future-internet.eu/activities/fp7-projects.html

[3] David D. Clark, “ Toward the design of a Future Internet”, Version 7, Oct 2009

[4] Th. Magedanz, S. Wahle “ Control framework design for Future Internet” Springer Eelktrotechnik & Informationstechnik(2009), p. 274-279

[5] GENI Security Architecture, Spiral 1 Draft 0.55, July 31th, 2009

[6] James Roberts, “The clean-slate approach to future Internet design: a survey of research initiatives” Ann. Telecommun. 2009, p 271-276

[7] http://www.nets-find.net/

[8] GENI: Global Environment for Network Innovations. http://www.geni.net/

[9] https://www.emulab.net/

[10] Nick Feamster, Lixin Gao and Jennifer Rexford, ”CABO: Concurrent Architectures are Better Than One”, http://www.netsfind.net/Funded/Cabo.php

[11] Thrasyvoulos Spyropoulos, Serge Fdida, Scott Kirkpatrick, “Future Internet: Fundamentals and Measurement” [Report of the COST Arcadia Future Internet Workshop]

[12] http://www.planet-lab.org/

[13] http://www.vini-veritas.net/

[14] http://www.networkworld.com/news/2008/062408-sloppy-virtualization.html

[15] http://www.networkworld.com/newsletters/nsm/2008/062308nsm1.html

[16] T.S. Eugene Ng, Alan L. Cox, “NeTS-FIND: Maestro: An Architecture for Network Control management”

[17] Alex C. Snoeren, Tadayoshi Kohno, Stefan Savage, Amin Vahdat, and Geoffrey M. Voelker. “Collaborative Research: NeTS—FIND:Privacy-Preserving Attribution and Provenance”

[18] David G. Andersen and Hari Balakrishnan and Nick Feamster and Teemu Koponen and Daekyeong Moon and cott Shenker, Accountable Internet Protocol (AIP), In Proc. ACM SIGCOMM, Aug 2008

[19] Mark Huang, Andy Bavier, Larry Peterson, PlanetFlow: Maintaining accountability for network services, ACM SIGOPS Operating Systems Review, v.40 n.1, January 2006

[20] K. Argyraki, P. Maniatis, O. Irzak, A. Subramanian, and S. Shenker, "Loss and Delay Accountability for the Internet”, ICNP 2007 Beijing, China.

[21] M. Shaw. Leveraging good intentions to reduce unwanted network traffic. In Proc. USENIX Steps to Reduce Unwanted Traffic on the Internet workshop, July 2006.

[22] Protecting America's Freedom in the Information Age. Markle Foundation, 2002. http://www.markle.org/downloadable_assets/nstf_full.pdf

[23] David Brin, ‘The Transparent Society’, Addison-Wesley, April 1998

[24] D. Weitzner, Beyond Secrecy: New Privacy Protection Strategies for Open Information Spaces, IEEE Internet Computing, Sept/Oct 2007.

[25] http://www.nsf.gov/news/news_summ.jsp?cntn_id=109589

[26] Weitzner, Abelson, Berners-Lee, et al., "Transparent Accountable Data Mining: New x Discretionary, rule-based access for the world wide web. In Elena Ferrari and Bhavani Thuraisingham, editors, Web and Information Security. IRM Press, 2006

[27] http://www.sparta.com/

[28] http://www.protogeni.net/trac/protogeni/wiki/Tutorial

[29] Weitzner, D., Hendler, J., Berners-Lee, T., & Connolly, D. Creating a policy-aware web

[30] http://www.geni.net/wp-content/uploads/2010/02/GENI_at_a_Glance_January_2010_Final-1.pdf, last accessed on 08-07-2010

[31] http://www.geni.net/wp-content/uploads/2010/02/Spiral1_Annual_Report.pdf, last accessed on 08-07-2010

[32] P. Ferguson and D. Senie. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. Internet Engineering Task Force, Jan. 1998. RFC 2267.

[33] M. Shaw. Leveraging good intentions to reduce unwanted network traffic. In Proc. X *Integrated Experimental Environment for Distributed Systems and Networks"*, appeared at OSDI 2002, December 2002

[34] White, Lepreau, Stoller, Ricci, Guruprasad, Newbold, Hibler, Barb, and Joglekar, *"An Integrated Experimental Environment for Distributed Systems and Networks"*, appeared at OSDI 2002, December 2002

[35] http://www.wisebed.eu/ (WISEBED: wireless sensor network testbed)

[36] http://www.shiratori.riec.tohoku.ac.jp/jgn.html (Japan Gigabit Network-JGN)

[37] http://www.nict.go.jp/section/index_e.html(NICT)

[38] http://www.protogeni.net/trac/protogeni

[39] http://groups.geni.net/geni/wiki/OldGPGDesignDocuments

[40] GENI Facility Security, GDD-06-23, Distributed Services working group, Draft work in progress (version 0.5)

[41] ProtoGENI CF Overview, 022709 GENI-SE-CF-ProtoGENIOver-01.4, February 27, 2009

[42] http://www.protogeni.net/trac/protogeni/wiki/RSpec

[43] Statement of work, University of Utah and Princeton University Proposal:" Exploring Federation of Testbeds with Diverse Models"

[44] GENI Security Architecture, Spiral 2 Draft 0.5, March 15th, 2010

[45] GENI: Towards Operational Security For GENI, Draft, GDD-06-10, July 2006

[46] http://ubuntuforums.org/

[47] Bishop, M. Gates, C. Frincke, D. Greitzer, F.L. , "AZALIA: an A to Z assessment of the likelihood of insider attack", Technologies for Homeland Security, 2009 IEEE, Page(s): 385 – 392

[48] Bishop, M., "Security Problems with the UNIX Operating System", version2, January 31, 1983[unpublished]

[49] Bishop, M., “Reflections on UNIX Vulnerabilities” , Computer Security Applications Conference, 2009. ACSAC 2009. Annual, Page(s): 161 - 184

[50] http://en.wikipedia.org/wiki/Secure_Shell#Security_issues

[51] http://www.cs.princeton.edu/~llp/arch_abridged.pdf

[52] http://en.wikipedia.org/wiki/Port_scanner

[53] Faber, T., Ricci, R.: Resources Description in GENI: Rspec model”, Second GENI Engineering Meeting, March, 2008

[54] http://www.protogeni.net/trac/protogeni/attachment/wiki/GEC8Tutorial/gec8tutorial-scripts.tar.gz

[55] http://www.protogeni.net/trac/protogeni/wiki

[56] http://www.protogeni.net/trac/protogeni/wiki/RSpecTutorial

[57] http://www.protogeni.net/trac/protogeni/wiki/RSpecExamples#tarball

[58] http://www.ic3.gov/media/annualreport/2009_IC3Report.pdf

[59] http://www.pcmag.com/article2/0,2817,2331225,00.asp, last accessed on 08-05-2010

[60] http://www.interlab.ait.ac.th/aintec09/GEC5-Newcomers-Session.pdf

[61] J. Owens, J. Matthews, A study of passwords and methods used in brute-force SSH attacks, Available on (last accessed on 06-10-2010): http://people.clarkson.edu/~owensjp/pubs/leet08.pdf

[62] Hochmuth, P. November 11, 2004. LinuxWorld. Linux is 'most breached' OS on the Net, security research firm says. Available at: http://www.linuxworld.com.au/index.php/id188808220;fp;2;fpid;1.

[63] SANS Institute. 2007. SANS Top-20 2007 Security Risks (2007 Annual Update). Available at: http://www.sans.org/top20/2007/

[64] Lemon, S. September 20, 2006. Computer World Security.Bruce Schneier: We are losing the security war. Available at: http://www.computerworld.com/s/article/9003477/Bruce_Schneier_We_are_losing_the_security_war

[65] Ramsbrock, D. Berthier, R. & Cukier, M. 2007. “Profiling Attacker Behavior Following SSH Compromises,” in Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, pp.119-124.

[66] S. Panjwani, S. Tan, K. Jarrin, and M. Cukier, “An Experimental Evaluation to Determine if Port Scans are Precursors to an Attack,” in Proceedings of the International Conference on Dependable Systems and Networks (DSN- 2005), Yokohama, Japan, June 28-July 1, 2005, pp. 602-611.

[67] Symantec December 17, 2007. Symantec Looks Back at the Internet Security Trends and Threats of 2007. Available at: http://www.symantec.com/about/news/resources/press_kits/detail.jsp?pkid=endofyear.

[68] http://openssh.org

[69] http://nmap.org

[70] http://en.wikipedia.org/wiki/SHA-1

[71] http://en.wikipedia.org/wiki/Secure_Shell

[72] http://linux.die.net/man/8/sshd

APPENDIX

Resource Specification File Details:

1. Simplest RSpec- one node without any specification of requested node

ANYNODE.XML

```
<rspec xmlns="http://protogeni.net/resources/rspec/0.1">

  <node virtual_id="mypc" virtualization_type="emulab-vnode" exclusive="1" />

      </rspec>
```

2. RSpec for getting shared nodes for experiments:

```
  <rspec xmlns="http://www.protogeni.net/resources/rspec/0.1"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:schemaLocation="http://www.protogeni.net/resources/rspec/0.1
          http://www.protogeni.net/resources/rspec/0.1/request.xsd"
        type="request" >
    <node component_manager_uuid="urn:publicid:IDN+emulab.net+authority+cm"
         virtual_id="shared1"
         virtualization_type="emulab-vnode"
         virtualization_subtype="emulab-openvz"
         exclusive="0">
      <node_type type_name="pcvm" type_slots="1"/>
      <interface virtual_id="virt-0"/>
      <interface virtual_id="control"/>
    </node>
    <node component_manager_uuid="urn:publicid:IDN+emulab.net+authority+cm"
         virtual_id="shared2"
         virtualization_type="emulab-vnode"
         virtualization_subtype="emulab-openvz"
         exclusive="0">
      <node_type type_name="pcvm" type_slots="1"/>
      <interface virtual_id="virt-0"/>
      <interface virtual_id="control"/>
    </node>
    <link virtual_id="link0" link_type="ethernet">
      <bandwidth>100000</bandwidth>
      <interface_ref virtual_node_id="shared2"
                 virtual_interface_id="virt-0"/>
      <interface_ref virtual_node_id="shared1"
                 virtual_interface_id="virt-0"/>
    </link>
  </rspec>
```

3. RSpec to install software on Remote ProtoGENI node:

```
   <?xml version="1.0" encoding="UTF-8"?>
   <rspec xmlns="http://www.protogeni.net/resources/rspec/0.1"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.protogeni.net/resources/rspec/0.1
http://www.protogeni.net/resources/rspec/0.1/request.xsd"
          type="request" >
     <node virtual_id="sh-node"
           virtualization_type="emulab-vnode"
           exclusive="1"
           tarfiles="/local http://www.emulab.net/downloads/protogeni-tests.tar.gz"
           startup_command="/bin/echo Example Text Here > /local/example.txt; /bin/cat
/local/example/text.txt >> /local/example.txt">
       <node_type type_name="pc" type_slots="1"/>
       <interface virtual_id="control"/>
     </node>
</rspec>
```

4. RSpec for requesting three nodes with three different OS types:

```
   <rspec xmlns="http://www.protogeni.net/resources/rspec/0.2" type="request">
    <node virtual_id="utah1"
          component_manager_uuid="urn:publicid:IDN+emulab.net+authority+cm"
          exclusive="1"
          virtualization_type="emulab-vnode"
          virtualization_subtype="raw">
       <interface virtual_id="virt1"/>
       <interface virtual_id="virt2"/>
       <interface virtual_id="control"/>
      <disk_image    name="urn:publicid:IDN+emulab.net+image+emulab-ops//FBSD62-STD" />
    </node>
    <node virtual_id="utah2"
          component_manager_uuid="urn:publicid:IDN+emulab.net+authority+cm"
          exclusive="1"
          virtualization_type="emulab-vnode"
          virtualization_subtype="raw">
       <interface virtual_id="virt1"/>
      <disk_image
         name="urn:publicid:IDN+emulab.net+image+emulab-ops//FEDORA8-OVZ-STD" />
    </node>

    <node virtual_id="utah3"
          component_manager_uuid="urn:publicid:IDN+emulab.net+authority+cm"
          exclusive="1"
          virtualization_type="emulab-vnode"
          virtualization_subtype="raw">
       <interface virtual_id="virt1"/>
      </node>
   -  <link virtual_id="link1">
       <interface_ref virtual_node_id="utah1"
                   virtual_interface_id="virt1" />    <interface_ref virtual_node_id="utah2"
                   virtual_interface_id="virt1" />
    </link>
    <link virtual_id="link2">
      <interface_ref virtual_node_id="utah1"
                   virtual_interface_id="virt2" />
      <interface_ref virtual_node_id="utah3"
                   virtual_interface_id="virt1" />
    </link>
</rspec>
```
[Utah1:FBSD62-STD, Utah2:FEDORA8-OVZ-STD, Utah3: default OS image]

5. RSpec for a specific OS on a particular machine: OSTYPE_ON_SPECIFIC_PC.XML

```
<?xml version="1.0" encoding="UTF-8"?>

<rspec xmlns="http://www.protogeni.net/resources/rspec/0.2"

    xmlns:xsi="http://www.w3.org/2001/XMLSchema-

instance"xsi:schemaLocation="http://www.protogeni.net/resources/rspec/0.2

http://www.protogeni.net/resources/rspec/0.2/request.xsd"

    type="request">

 <node

component_uuid="urn:publicid:IDN+emulab.net+node+pc175"component_manager_uuid="urn:publicid:

IDN+emulab.net+authority+cm"    virtual_id="pc175"

     virtualization_type="emulab-vnode"

     exclusive="1">

 <node_type type_name="pc" type_slots="1"/>

 <disk_image  name="urn:publicid:IDN+emulab.net+image+emulab-ops//FBSD72-STD" />

 </node>

</rspec>
```

6. RSpec for a specific OS type: *OSTEST.XML*

```
   <rspec xmlns="http://protogeni.net/resources/rspec/0.2">

      <node virtual_id="geni1" exclusive="1" virtualization_type="emulab-vnode"

      virtualization_subtype="raw">

 <disk_image name="urn:publicid:IDN+emulab.net+image+emulab-ops//FBSD62-STD" />

     </node>  </rspec>
```

updatesliver.py details

// created a sliver testslice with shrspec1.xml

```
sshalini@ubuntu:~/protogeni/test$ ./createsliver.py -n testslice shrspec1.xml

Got my SA credential

No such slice registered here:Creating new slice called testslice

New slice created

Creating the Sliver ...

Created the sliver

<rspec xmlns="http://protogeni.net/resources/rspec/0.1" valid_until="2010-07-20T20:16:41">

<node virtual_id="geni1" virtualization_type="raw" exclusive="1"

component_urn="urn:publicid:IDN+emulab.net+node+pc50" component_uuid="de982479-

773e-102b-8eb4-001143e453fe"

component_manager_urn="urn:publicid:IDN+emulab.net+authority+cm"

component_manager_uuid="28a10955-aa00-11dd-ad1f-001143e453fe" sliver_uuid="de982479-

773e-102b-8eb4-001143e453fe" hostname="pc50.emulab.net" sshdport="22"

sliver_urn="urn:publicid:IDN+emulab.net+sliver+14650">

<interface virtual_id="virt0" component_id="eth3"/>

<services><login authentication="ssh-keys" hostname="pc50.emulab.net"

port="22"/></services></node>

<node virtual_id="geni2" virtualization_type="raw" exclusive="1"

component_urn="urn:publicid:IDN+emulab.net+node+pc68" component_uuid="de988e65-

773e-102b-8eb4-001143e453fe"

component_manager_urn="urn:publicid:IDN+emulab.net+authority+cm"
```

```
component_manager_uuid="28a10955-aa00-11dd-ad1f-001143e453fe" sliver_uuid="de988e65-
773e-102b-8eb4-001143e453fe" hostname="pc68.emulab.net" sshdport="22"
sliver_urn="urn:publicid:IDN+emulab.net+sliver+14651">
<interface virtual_id="virt0" component_id="eth3"/>
<services><login authentication="ssh-keys" hostname="pc68.emulab.net"
port="22"/></services></node>
<link virtual_id="link0" sliver_uuid="c61ffd61-9409-11df-ad83-001143e453fe"
sliver_urn="urn:publicid:IDN+emulab.net+sliver+14652" vlantag="74">
<interface_ref virtual_interface_id="virt0" virtual_node_id="geni1" sliver_uuid="c9391289-
9409-11df-ad83-001143e453fe"
component_urn="urn:publicid:IDN+emulab.net+interface+pc50:eth3"
sliver_urn="urn:publicid:IDN+emulab.net+sliver+14653" MAC="0002b3237915"
IP="10.10.1.1"/>
<interface_ref virtual_interface_id="virt0" virtual_node_id="geni2" sliver_uuid="d4167140-
9409-11df-ad83-001143e453fe"
component_urn="urn:publicid:IDN+emulab.net+interface+pc68:eth3"
sliver_urn="urn:publicid:IDN+emulab.net+sliver+14654" MAC="0002b33f75c1"
IP="10.10.1.2"/>
</link>
</rspec>
```

// sliver testslice was updated as per shrspec2.xml (node_id names were differenet)

```
sshalini@ubuntu:~/protogeni/test$ ./updatesliver.py -n testslice shrspec2.xml
Got my SA credential
```

```
Asking for slice credential for testslice

Got the slice credential

Resolving the slice at the CM

{'urn': 'urn:publicid:IDN+emulab.net+slice+testslice', 'sliver_urn':

'urn:publicid:IDN+emulab.net+sliver+14649'}

Asking for sliver credential

Got the sliver credential

Updating the Sliver ...

Updated the sliver, got a new ticket
```

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<redeem_before>2010-07-20T14:35:40</redeem_before>

  <rspec xmlns="http://protogeni.net/resources/rspec/0.1">

<node virtual_id="node1" virtualization_type="raw" exclusive="1"

component_urn="urn:publicid:IDN+emulab.net+node+pc50" component_uuid="de982479-

773e-102b-8eb4-001143e453fe"

component_manager_urn="urn:publicid:IDN+emulab.net+authority+cm"

component_manager_uuid="28a10955-aa00-11dd-ad1f-001143e453fe">

<interface virtual_id="virt0" component_id="eth3"/>

</node>

<node virtual_id="node2" virtualization_type="raw" exclusive="1"

component_urn="urn:publicid:IDN+emulab.net+node+pc68" component_uuid="de988e65-

773e-102b-8eb4-001143e453fe"
```

```
component_manager_urn="urn:publicid:IDN+emulab.net+authority+cm"

component_manager_uuid="28a10955-aa00-11dd-ad1f-001143e453fe">

<interface virtual_id="virt0" component_id="eth3"/>

</node>

<link virtual_id="link0">

<interface_ref virtual_interface_id="virt0" virtual_node_id="node1"/> // node_id as per

shrspec2.xml

<interface_ref virtual_interface_id="virt0" virtual_node_id="node2"/>

</link>

…..

// sliver status same as before: 2 nodes as got when first sliver was created with shrspec1.xml

sshalini@ubuntu:~/protogeni/test$ ./sliverstatus.py -n testslice

Got my SA credential. Looking for slice ...

Found the slice, asking for a credential ...

Got the slice credential, asking for a sliver credential ...

Got the sliver credential, asking for sliver status

{'status': 'ready', 'state': 'started', 'details': {'urn:publicid:IDN+emulab.net+sliver+14651':

{'status': 'ready', 'state': 'started', 'component_urn': 'urn:publicid:IDN+emulab.net+node+pc68',

'error': ''}, 'urn:publicid:IDN+emulab.net+sliver+14650': {'status': 'ready', 'state': 'started',

'component_urn': 'urn:publicid:IDN+emulab.net+node+pc50', 'error': ''}}}

sshalini@ubuntu:~/protogeni/test$ ssh shail01@pc68.emulab.net

The authenticity of host 'pc68.emulab.net (155.98.36.68)' can't be established.

RSA key fingerprint is 6d:1d:76:53:a5:25:99:39:e2:89:ea:b0:99:e3:d3:b9.
```

```
Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added 'pc68.emulab.net,155.98.36.68' (RSA) to the list of known hosts.

[shail01@geni2 ~]$ logout // geni2 was node_id,specified in shrspec1.xml

Connection to pc68.emulab.net closed.

sshalini@ubuntu:~/protogeni/test$ ./renewsliver.py -n testslice 5000

Got my SA credential

Found the slice, asking for a credential ...

Got the slice credential, renewing the slice at the SA ...

Renewed the slice, asking for slice credential again

Got the slice credential, renewing the sliver

Sliver has been renewed until 20100724T03:54:59

//again updated to get a single node with anynode.xml

sshalini@ubuntu:~/protogeni/test$ ./updatesliver.py -n testslice anynode.xml

Got my SA credential

Asking for slice credential for testslice

Got the slice credential

Resolving the slice at the CM

{'urn': 'urn:publicid:IDN+emulab.net+slice+testslice', 'sliver_urn':

'urn:publicid:IDN+emulab.net+sliver+14649'}

Asking for sliver credential

Got the sliver credential

Updating the Sliver ...

Updated the sliver, got a new ticket
```

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<signed-credential xmlns:xsi="http://www.w3.org/2001/...

..

// showing resource as only one pc asked to update through anynode.xml


<redeem_before>2010-07-20T21:49:55</redeem_before>

  <rspec xmlns="http://protogeni.net/resources/rspec/0.1">

 <node virtual_id="mypc" virtualization_type="raw" exclusive="1"

component_urn="urn:publicid:IDN+emulab.net+node+pc68" component_uuid="de988e65-

773e-102b-8eb4-001143e453fe"

component_manager_urn="urn:publicid:IDN+emulab.net+authority+cm"

component_manager_uuid="28a10955-aa00-11dd-ad1f-001143e453fe">

 </node>

..

// sliver status same as before: 2 nodes as got when first sliver was created with shrspec1.xml

sshalini@ubuntu:~/protogeni/test$ ./sliverstatus.py -n testslice

Got my SA credential. Looking for slice ...

Found the slice, asking for a credential ...

Got the slice credential, asking for a sliver credential ...

Got the sliver credential, asking for sliver status

{'status': 'ready', 'state': 'started', 'details': {'urn:publicid:IDN+emulab.net+sliver+14651':

{'status': 'ready', 'state': 'started', 'component_urn': 'urn:publicid:IDN+emulab.net+node+pc68',

 'error': ''},
 'urn:publicid:IDN+emulab.net+sliver+14650': {'status': 'ready', 'state': 'started',
  'component_urn': 'urn:publicid:IDN+emulab.net+node+pc50', 'error': ''}}}
sshalini@ubuntu:~/protogeni/test$ ./redeemticket.py -n testslice 200
Got my SA credential
Asking for slice credential for testslice
Got the slice credential
Resolving the slice at the CM
{'urn': 'urn:publicid:IDN+emulab.net+slice+testslice', 'ticket_urn':
'urn:publicid:IDN+emulab.net+ticket+62210', 'sliver_urn':
'urn:publicid:IDN+emulab.net+sliver+14649'}
Asking for a copy of the ticket
Got the ticket
Asking for sliver credential
Got the sliver credential
Redeeming the ticket
Could not create GeniSliver object for mypc: Could not redeem the ticket

sshalini@ubuntu:~/protogeni/test$ ./sliverstatus.py -n testslice
Got my SA credential. Looking for slice ...
Found the slice, asking for a credential ...
Got the slice credential, asking for a sliver credential ...
Got the sliver credential, asking for sliver status
Could not get sliver status
```

```
sshalini@ubuntu:~/protogeni/test$ ./sliveraction.py -n testslice restart

Got my SA credential. Looking for slice ...

Found the slice, asking for a credential ...

Got the slice credential, asking for a sliver credential ...

Got the sliver credential, calling RestartSliver on the sliver

Could not start sliver
```

```
sshalini@ubuntu:~/protogeni/test$ ./sliveraction.py -n testslice start

Got my SA credential. Looking for slice ...

Found the slice, asking for a credential ...

Got the slice credential, asking for a sliver credential ...

Got the sliver credential, calling StartSliver on the sliver

Could not start sliver
```

Client-Server Program files:

a. *Client Program:*

```
#include <sys/types.h>
#include <sys/socket.h> /* for socket(), connect(), send(), and recv() */
#include <netinet/in.h>
#include <arpa/inet.h>  /* for sockaddr_in and inet_addr() */
#include <stdio.h>      /* for printf() and fprintf() */
#include <stdlib.h>     /* for atoi() and exit() */
#include <string.h>     /* for memset() */
#include <unistd.h>     /* for close() */

#define RCVBUFSIZE 32   /* Size of receive buffer */

void DieWithError(char *errorMessage);  /* Error handling function */
unsigned long ResolveName (char name[]);

int main(int argc, char *argv[])
{
   int sock;                        /* Socket descriptor */
```

```
struct sockaddr_in echoServAddr; /* Echo server address */
unsigned short echoServPort;     /* Echo server port */
char *servIP;                    /* Server IP address (dotted quad) */
char *echoString;                /* String to send to echo server */
char echoBuffer[RCVBUFSIZE];     /* Buffer for echo string */
unsigned int echoStringLen;      /* Length of string to echo */
int bytesRcvd, totalBytesRcvd;   /* Bytes read in single recv()
                                    and total bytes read */

if ((argc < 3) || (argc > 4))    /* Test for correct number of arguments */
{
   fprintf(stderr, "Usage: %s <Server IP or Hostname> <Echo Word> [<Echo Port>]\n",
           argv[0]);
   exit(1);
}

servIP = argv[1];             /* First arg: server IP address (dotted quad) */
echoString = argv[2];         /* Second arg: string to echo */

if (argc == 4)
    echoServPort = atoi(argv[3]); /* Use given port, if any */
else
    echoServPort = 7;  /* 7 is the well-known port for the echo service */

/* Create a reliable, stream socket using TCP */
if ((sock = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP)) < 0)
    DieWithError("socket() failed");

/* Construct the server address structure */
memset(&echoServAddr, 0, sizeof(echoServAddr));     /* Zero out structure */
echoServAddr.sin_family      = AF_INET;             /* Internet address family */
echoServAddr.sin_addr.s_addr = ResolveName(servIP);   /* Server IP address or hostname */
echoServAddr.sin_port        = htons(echoServPort); /* Server port */

/* Establish the connection to the echo server */
if (connect(sock, (struct sockaddr *) &echoServAddr, sizeof(echoServAddr)) < 0)
    DieWithError("connect() failed");

echoStringLen = strlen(echoString);          /* Determine input length */

/* Send the string to the server */
if (send(sock, echoString, echoStringLen, 0) != echoStringLen)
    DieWithError("send() sent a different number of bytes than expected");

/* Receive the same string back from the server */
totalBytesRcvd = 0;
printf("Received: ");                /* Setup to print the echoed string */
while (totalBytesRcvd < echoStringLen)
{
    /* Receive up to the buffer size (minus 1 to leave space for
       a null terminator) bytes from the sender */
    if ((bytesRcvd = recv(sock, echoBuffer, RCVBUFSIZE - 1, 0)) <= 0)
        DieWithError("recv() failed or connection closed prematurely");
    totalBytesRcvd += bytesRcvd;   /* Keep tally of total bytes */
    echoBuffer[bytesRcvd] = '\0';  /* Terminate the string! */
    printf(echoBuffer);            /* Print the echo buffer */
    }

    printf("\n");    /* Print a final linefeed */

    close(sock);
    exit(0);
}
```

b. Server Program

```
#include <sys/types.h>
#include <sys/socket.h> /* for socket(), connect(), send(), and recv() */
#include <netinet/in.h>
#include <arpa/inet.h>  /* for sockaddr_in and inet_addr() */
#include <stdio.h>      /* for printf() and fprintf() */
#include <stdlib.h>     /* for atoi() and exit() */
#include <string.h>     /* for memset() */
#include <unistd.h>     /* for close() */


#define MAXPENDING 5    /* Maximum outstanding connection requests */

void DieWithError(char *errorMessage);  /* Error handling function */
void HandleTCPClient(int clntSocket);   /* TCP client handling function */

int main(int argc, char *argv[])
{
    int servSock;                    /* Socket descriptor for server */
    int clntSock;                    /* Socket descriptor for client */
    struct sockaddr_in echoServAddr; /* Local address */
    struct sockaddr_in echoClntAddr; /* Client address */
    unsigned short echoServPort;     /* Server port */
    unsigned int clntLen;            /* Length of client address data structure */

    if (argc != 2)     /* Test for correct number of arguments */
    {
        fprintf(stderr, "Usage:  %s <Server Port>\n", argv[0]);
        exit(1);
    }

    echoServPort = atoi(argv[1]);  /* First arg:  local port */

    /* Create socket for incoming connections */
    if ((servSock = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP)) < 0)
        DieWithError("socket() failed");

    /* Construct local address structure */
    memset(&echoServAddr, 0, sizeof(echoServAddr));   /* Zero out structure */
    echoServAddr.sin_family = AF_INET;                /* Internet address family */
    echoServAddr.sin_addr.s_addr = htonl(INADDR_ANY); /* Any incoming interface */
    echoServAddr.sin_port = htons(echoServPort);      /* Local port */

    /* Bind to the local address */
    if (bind(servSock, (struct sockaddr *) &echoServAddr, sizeof(echoServAddr)) < 0)
        DieWithError("bind() failed");
    /* Mark the socket so it will listen for incoming connections */
    if (listen(servSock, MAXPENDING) < 0)
        DieWithError("listen() failed");

    for (;;) /* Run forever */
    {
        /* Set the size of the in-out parameter */
        clntLen = sizeof(echoClntAddr);

        /* Wait for a client to connect */
        if ((clntSock = accept(servSock, (struct sockaddr *) &echoClntAddr,
                               &clntLen)) < 0)
            DieWithError("accept() failed");

        /* clntSock is connected to a client! */

        printf("Handling client %s\n", inet_ntoa(echoClntAddr.sin_addr));

        HandleTCPClient(clntSock);
    }
    /* NOT REACHED */
}
```

c. Supporting files

i. DieWithError

```
#include <stdio.h>  /* for perror() */
#include <stdlib.h> /* for exit() */

void DieWithError(char *errorMessage)
{
    perror(errorMessage);
    exit(1);
}
```

ii. HandleTCPClient

```
#include <stdio.h>      /* for printf() and fprintf() */
#include <sys/socket.h> /* for recv() and send() */
#include <unistd.h>     /* for close() */

#define RCVBUFSIZE 32   /* Size of receive buffer */

void DieWithError(char *errorMessage);  /* Error handling function */
```

```c
void HandleTCPClient(int clntSocket)
{
    char echoBuffer[RCVBUFSIZE];        /* Buffer for echo string */
    int recvMsgSize;                    /* Size of received message */

    /* Receive message from client */
    if ((recvMsgSize = recv(clntSocket, echoBuffer, RCVBUFSIZE, 0)) < 0)
        DieWithError("recv() failed");

    /* Send received string and receive again until end of transmission */
    while (recvMsgSize > 0)      /* zero indicates end of transmission */
    {
        /* Echo message back to client */
        if (send(clntSocket, echoBuffer, recvMsgSize, 0) != recvMsgSize)
            DieWithError("send() failed");

        /* See if there is more data to receive */
        if ((recvMsgSize = recv(clntSocket, echoBuffer, RCVBUFSIZE, 0)) < 0)
            DieWithError("recv() failed");
    }

    close(clntSocket);    /* Close client socket */
}
```

iii. ResolveName

```c
#include <stdio.h>      /* for fprintf() */
#include <netdb.h>      /* for gethostbyname() */
#include <stdlib.h>     /* for exit() */

unsigned long ResolveName(char name[])
{
    struct hostent *host;           /* Structure containing host information */

    if ((host = gethostbyname(name)) == NULL)
    {
        fprintf(stderr, "gethostbyname() failed");
        exit(1);
    }

    /* Return the binary, network byte ordered address */
    return *((unsigned long *) host->h_addr_list[0]);
}
```

Ping details of wireless nodes pcwf1, pcwf3, pcwf5, and pcwf7

no. of pkts.	Environment	node-id	node-address	rtt (ms)					loss of packets(no.)	loss of packets(%)
				min	max	avg	mdev	pipe		
10	one-by one	pcwf1	10.1.1.2	0.653	2.021	7.702	2.211	2	0	0
		pcwf3	10.1.1.3	1.564	4.033	9.108	2.021	2	0	0
		pcwf5	10.1.1.5	1.996	5.469	20.998	5.577	2	0	0
		pcwf7	10.1.1.4	0.717	1.883	3.782	0.709	2	0	0
10	all together	pcwf1	10.1.1.2	0.663	1.517	6.002	1.62	2	0	0
		pcwf3	10.1.1.3	1.129	1.689	3.709	0.711	2	0	0
		pcwf5	10.1.1.5	2.383	4.007	11.73	2.697	2	0	0
		pcwf7	10.1.1.4	1.75	2.386	7.467	1.696	2	0	0
50	one-by-one	pcwf1		0.536	1.307	4.381	0.789	2	2	4
		pcwf3		1.444	2.355	5.918	1.055	2	0	0
		pcwf5		1.394	2.616	8.941	1.449	2	2	4
		pcwf7		0.576	1.233	5.526	0.749	2	0	0
				0.9875	1.8778	6.1915	1.0105			
50	all together	pcwf1		0.522	1.24	9.204	1.362	2		
		pcwf3		1.51	2.749	7.753	1.436	2		
		pcwf5		1.551	2.452	7.472	1.113	2	1	2
		pcwf7		0.642	1.529	5.168	0.819	2		
				1.05625	1.9925	7.39925	1.1825			
100	one-by-one	pcwf1		0.554	1.426	8.732	1.137	2		
		pcwf3		1.397	2.49	9.037	1.439	2		
		pcwf5		1.52	2.878	8.805	1.503	2	2	2
		pcwf7		0.531	1.185	10.902	1.234	2		
	Average of all four in each case			1.0005	1.9948	9.369	1.32825			
100	all together	pcwf1		0.645	1.216	4.992	0.855	2	1	1
		pcwf3		1.508	2.412	11.226	1.318	2		
		pcwf5		1.534	2.397	11.366	1.578	2		
		pcwf7		0.572	1.07	4.487	0.788	2		
				1.06475	1.7738	8.01775	1.13475			
200	one-by-one	pcwf1		0.556	1.261	6.527	0.963	2	2	1
		pcwf3		1.509	2.441	6.805	1.032	2	1	

------ continue										
		pcwf5		1.481	2.237	11.05	1.212	2	1	
		pcwf7		0.489	1.095	10.304	1.203	2	0	
				1.00875	1.7585	8.6715	1.1025			
200	all together	pcwf1		0.458	1.247	8.796	1.114	2		
		pcwf3		1.252	2.604	16.242	1.687	2		
		pcwf5		1.34	2.277	8.263	1.058	2	1	
		pcwf7		0.524	14.148	1810.63	139.659	3		
				0.8935	5.069	**460.983**	35.8795			
300	one-by-one	pcwf1		0.623	1.416	13.992	1.199	2	2	
		pcwf3		1.391	2.564	12.639	1.635	2	2	
		pcwf5		1.301	2.603	11.299	1.658	2	2	
		pcwf7		0.537	1.043	12.76	1.216	2	0	
				0.963	1.9065	12.6725	1.427			
300	all together	pcwf1		0.618	1.375	8.256	1.066	2	2	
		pcwf3		1.108	5.28	787.506	45.486	2	3	1
		pcwf5		1.388	2.313	11.101	1.245	2	3	1
		pcwf7		0.454	4.401	1010.22	58.182	3	0	
				0.892	3.3423	**454.272**	26.4948			
400	one-by-one	pcwf1		0.636	1.449	15.625	1.256	2	1	
		pcwf3		1.404	3.939	407.391	20.36	2	4	1
		pcwf5		1.279	5.003	1011.78	50.553	3	2	
		pcwf7		0.369	1.013	12.264	1.316	2	0	
				0.922	2.851	**361.766**	18.3713			
400	all together	pcwf1		0.612	1.204	8.725	0.922	2	3	0

-----continue										
		pcwf3		1.262	5.154	1023.63	51.135	3	2	0
		pcwf5		1.252	2.399	12.641	1.387	2	1	0
		pcwf7		0.418	8.488	3010.27	150.657	5	2	0
				0.886	4.3113	**1013.82**	51.0253			
500	one-by-one	pcwf1		0.584	1.144	5.141	0.76	2	4	0
		pcwf3		0.993	5.312	1005.67	48.03	2	4	0
		pcwf5		1.272	2.364	14.716	1.469	2	4	0
		pcwf7		0.457	1.145	10.196	1.201	2	0	0
				0.8265	2.4913	258.93	12.865			
500	all together	pcwf1		0.433	3.422	1004.46	45.117	2	6	1
		pcwf3		1.383	2.77	12.429	1.755	2	4	0
		pcwf5		1.22	8.338	2883.3	129.102	4	3	0
		pcwf7		0.417	0.943	10.879	0.924	2	0	0
				0.86325	3.8683	**977.766**	44.2245			

Table 4.2: Wireless communication between Emulab resources