Evaluation of Security Vulnerabilities by Using ProtoGENI as a Launchpad

Dawei Li, Xiaoyan Hong, Jason Bowman
Department of Computer Science
University of Alabama, Tuscaloosa, AL 35487

***Abstract*—In this paper we analyze the security architecture of ProtoGENI. ProtoGENI is a prototype control framework implementation of GENI (Global Environment for Network Innovations). We perform a variety of experiments in an effort to identify potential vulnerabilities presented in the current implementation. We classify our attacks into three types: data plane to data plane, data plane to control plane, and data plane to Internet. Our results indicate the potential for a breach of confidentiality and availability internally within ProtoGENI, as well as risks to external Internet. We make suggestions outlining possible defense strategies to improve ProtoGENI security and aid in future development.** [1]

***Index Terms*—ProtoGENI, GENI security, vulnerability, GENI experiments**

I. Introduction

The Global Environment for Network Innovations (GENI) is a virtual laboratory for at-scale networking experimentation [2][7]. This global network research testbed provides computing and networking resources (PCs, routers, switchers, wireless devices, OpenFlow etc.) that are geographically distributed and federated through control frameworks. GENI allows networking experimentation to be conducted at Internet scale and to be able to use a large variety of networking technologies both current and forthcoming. One critical issue for such an at-scale infrastructure is security. It is extremely challenging to achieve security goals due to the many unique features GENI has, which include distributed ownership and users groups, deep programmability of the infrastructure resources, super flexibility of congurability of the virtualized resources, large scale connectivity to the Internet and large geographic span, etc. In addition, the vast variety of research experiments that could conduct at GENI may produce network behaviors in patterns that make the detection of an anomaly difficult.

The primary goal of GENI security is to avoid abuse of services to conduct illegal activities or as a launchpad for attacks, and to ensure that availability of services is not compromised by attacks [3]. Yet, there is more to understand the security requirements of GENI (in our case, ProtoGENI). Our project takes an experimenter's approach to identify potential system vulnerabilities present in the current implementation and provide the development team with possible solutions to those vulnerabilities [5]. It is our hope that our experiments will assist in building a more secure research infrastructure. Our experimentation was performed in conjunction with related personnel and under the supervision of applicable authorities, and potentially harmful experiments were limited to our own testbed slices and machines.

Our current work focuses on the ProtoGENI control framework [4]. ProtoGENI is built on the network research infrastructure of Emulab at Utah [1]. Emulab, and by association ProtoGENI, provides researchers a wide range of networking environments under which they can control conditions and securely conduct repeatable research experiments. ProtoGENI can be regarded as both a hardware facility providing computing and networking resources and as a software application defining a control framework which dictates policies for user authentication, resource allocation, and communication between different parties.

In this paper, we first analyze the potential security issues of ProtoGENI based on its control framework architecture. Then we describe the network experiments that explore the use of GENI resources as a launchpad for attacks [2]. We classify three types of attacks: data plane to data plane attack, data plane to control plane attack, and data plane to Internet attack. The results of our experiments show that the risk can be defined in either of two major categories. First, we are able to compromise the confidentiality and availability of other experimenters trying to utilize the resources of ProtoGENI. Secondly, ProtoGENI resources can be used as a potential source to launch attacks against global Internet users. The GENI security goal can be compromised with our preliminary experiments. The contribution of this work is to examine known vulnerabilities that have affected other systems and networks and to see if they are effective against ProtoGENI. We report these findings to corresponding GENI development teams and suggest possible defense strategies.

Our paper will be organized to introduce ProtoGENI (in Section II), general analysis on ProtoGENI security (Section III), and the attacking experiments as well as possible defense strategies (Section IV). At last we conclude the paper (Section V).

II. Background

A. ProtoGENI Control Framework

ProtoGENI implements and deploys the GENI Control Framework structure. Here are the key components of ProtoGENI facility and functioning software entities. They describe the management and the principles of the usage of GENI and ProtoGENI.

- ClearingHouse (CH): Center for registration;
- Component Manager (CM): Resource provider, managing the resources located at a particular location;
- Slice: Container for resources which can cross many CMs;
- Slice Authority (SA): Managing the slices, authenticating users to slices;
- Sliver: Computing resources granted to a slice;
- RSpec: Resource specification, used for advertising, requesting and describing the resources;

[1]This work is supported in part by BBN/NSF contract project 1783.

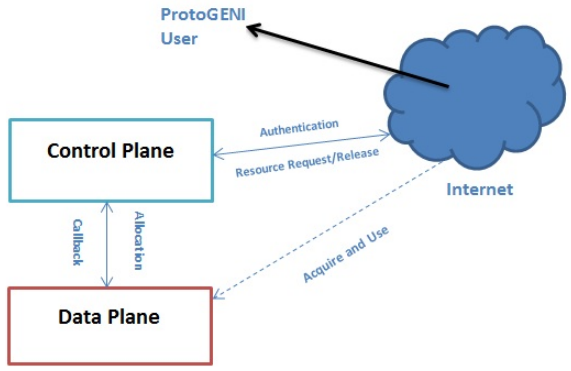

Fig. 1. ProtoGENI Control Plane and Data Plane

- Vnode: Virtual node, a node in the current sliver which shares the resources on physical server with other slices using virtualization. Current ProtoGENI realizes Vnode through OpenVZ.

B. Related Work

The DETER testbed is an Emulab-based infrastructure designed for medium-scale repeatable experiments in computer security [8]. It includes several mechanisms to deal with threats such as monitoring control network, placing firewalls, and physically separating experimental network from control network to prevent packets from being routed outside one's experiment. However, in the project, not enough testing cases about these mechanisms were included. The GENI security architecture Toolkit project [3][9] defines the threat model of GENI defines three broad classes of attacks. They consider the possible threats caused by the control framework running malicious or compromised software. But the threat model is only a high level description and the implementation of the prototype control framework would be different and need to be further explored.

III. ProtoGENI Security Analysis

A. ProtoGENI Resource Connection

1) Control Plane and Data Plane: ProtoGENI facility can be viewed as two separate planes: Control Plane and Data (Experimental) Plane. The data plane contains users' slices and is where the experiments are. The experiments can be configured for the network topology that they define through with RSpec. The control plane is a separate network that configures and interacts with the data plane to support the experiments. It also allows users to access the slices from outside of GENI, usually through Internet connection **with SSH**. The architecture is shown in Figure 1. In this architecture, all the experimental nodes in data plane are connected to a control-router in a control LAN with publicly accessible IP addresses.

2) Security Concerns: Connecting a large number of available nodes within a single LAN has potential problems. Once a malicious user obtains the control of one experiment node which connects to the control-router that links other experiment nodes to form the same LAN sharing , he can easily launch attacks disturbing the connection of another experiment node to the control router and even to the Internet. This kind of attacks includes LAN-specific attacks such as ARP poisoning attack [10].

Meanwhile, the ProtoGENI node has a public IP address for users to conveniently access the node through Internet connection. The potential vulnerability is quite obvious. The threat can be in two directions; on the one hand, a malicious user can use ProtoGENI experiment resources with the huge number of available computing resources and high up to 1Gbps bandwidth as a launchpad to harm the existing Internet user; on the other hand, attackers in the Internet will easily have these nodes as attacking targets. With the public IP address, attackers can do a port scan on the target node and perform further attacks based on the scan result.

B. ProtoGENI Wireless Nodes Distribution

The wireless node in ProtoGENI has a real wireless network interface which makes the attacks such as packet sniffing or spoofing to the targeted wireless interface card possible. However, the feasibility of wireless attack needs to consider two other facts.

The first fact is the location of wireless nodes in another running experiment slice. The attackers will want to use a node close to an experiment node to have a better performance. The ProtoGENI provides a map showing the physical locations of the wireless nodes and their availabilities. This provides normal users intuitive and clear view for selecting experimental nodes. However the attacker can also easily find his desired node for launching attacks. Another attraction for a malicious user is named resource scramble, i.e. an attacker requests all the nodes around a running wireless node so that the normal user cannot get a second wireless node within the radio range of his running wireless node.

Another fact is the wireless channel used in the running wireless experiment. In Emulab website, it provides a list showing the occupied channels. The web suggests new wireless experiment to avoid mutual interference and indicating by using other channels. The malicious user can easily have this information. Knowing the victim wireless channel can save the attackers trouble of adjusting its own channel to match the victims channel.

C. ProtoGENI Virtualization Technology

One important feature of GENI is virtualization. It means whenever possible the GENI resources are shared among as many users as possible while meeting their computing and networking requirement. However this virtualized environment should be carefully developed and any bugs or restrictions inside this environment may lead to an exposed entry for another user whose slice shares the same physical node.

Another key consideration of virtualization is to make sure the equitable distribution of computing and network resources such as processor power, memory and bandwidth. Any defects of the adopted virtualization technology will bring a user the potential to interrupt the experiments of the others with whom he shares the resources, be it on purpose or not.

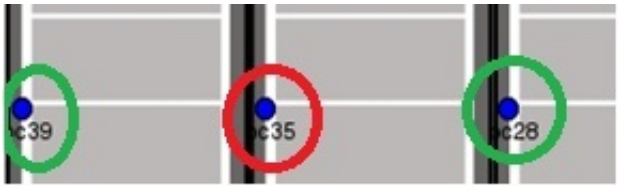

Fig. 2. Physical Location of Wireless Nodes

IV. Security Experiments and Results

In this section, we perform a series of experiments to verify the security issues analyzed in the previous section. All the experiments act as an intruder who has obtained the authority to create and run experiments in ProtoGENI as a normal user. Three types of attacks will be performed: data plane to data plane attack, i.e. an attacker uses his experimental resource to attack another running experiment; data plane to control plane attack, i.e. an attacker uses his experimental resource to attack the control network; and third, data plane to the Internet attack, i.e. an attack uses his experimental resource to attack hosts or servers residing in the Internet.

A. Experiments Methods and Tools

We use common network testing tools for our experiments, for examples, *ping* and *iperf*. We also use a software named *netwox* [11], an open source network tool set, to perform network attacks like packet sniffing and spoofing. We use *stress* as a system workload generator to impose a configurable amount of CPU, memory I/O and disk stress on the system in our experiments relating to the virtualization issue.

B. Data Plane to Data Plane Attack

Data Plane to Data Plane attack is the kind of attacks launched by a malicious ProtoGENI user to attack other users' running experiments. We tackle security in two directions, one in wireless network experiment and the other in virtualization.

1) Wireless Experiments: We perform two successive attacking experiments *Packet Sniffing* and *Packet Spoofing*.

a. Packet Sniffing:

Experiment Description: We perform the following experiment to test the feasibility of sniffing a packet from the air. In this experiment, two slices *experiment1* and *experiment2* are created. The slice *experiment1* is a normal user slice with two wireless nodes named *nodew1* and *nodew2* forming a topology of $nodew1 - - nodew2$. The two nodes are installed with Iperf. The slice *experiment2* acts as a launchpad for an attacker with only one node named *nodew1*. All the nodes have a wireless 802.11g interface and *netwox* is installed on the attacker's node. The three nodes are located in the following physical positions (Figure 2). Both slices choose channel 14.

First, in the normal slice *experiment1*, *iperf* server runs on *nodew2* and *iperf* client runs on *nodew1* to connect to the iperf server. Then, malicious *nodew1* of *experiment2* uses the No. 7 tool of netwox to sniff the TCP packets generated in the *experiment1*. The following command is used to sniff the TCP packets in the air:

$$netwox\ 7\ -d\ ath0\ -t\ ,$$

where *ath0* is the wireless interface and *-t* is used to sniff TCP packets.

Experiment Result and Analysis: The packet sniffed is shown in Figure 3. In this sniffed packet, we obtain information about both the server and the client, including IP addresses, MAC addresses and time stamps of the transmission which can be used for further exploration.

```
Ethernet
| 00:17:9A:C3:65:24->00:17:9A:08:C1:79 type:0x0800
IP
|version|  ihl  |      tos      |            totlen
|___4___|___5___|____0x00=0_____|___________0x0054=84___________|
|              id               |r|D|M|      offsetfrag
|_________0x10AC=4268___________|0|0|0|_______0x0000=0_________|
|       ttl       |   protocol    |            checksum
|____0x40=64____|____0x01=1_____|___________0x53F7_____________|
|                          source
|_____________________________10.1.1.2_____________________________|
|                        destination
|_____________________________10.1.1.3_____________________________|
```

Fig. 3. Part of the TCP Packet Sniffed

b. Packet Spoofing:

Experiment Description: By sniffing packets from a wireless communication, we can get both MAC address and IP address of the communication pairs. It will be a potential vulnerability because an attacker can launch the ARP cache poisoning attack by sending spoofed ARP packet to a victim host with faked IP-MAC addresses mapping about another host. Here we design a demo experiment showing how a ProtoGENI user can use ARP cache poisoning to attack another experiment slice with the netwox tool sets.

From the sniffed packet above, we know the source's IP address is 10.1.1.2 and MAC address is $00:17:9A:C3:65:24$; the destination's IP address is 10.1.1.3 and MAC address is $00:17:9A:08:C1:79$. We validate this information by checking the ARP table in *nodew1* of *experiment1*, we observe that the MAC address for *nodew2* (10.1.1.3) is $00:17:9A:08:C1:79$, which is the same as shown in the sniffed packet.

In the malicious *nodew1* of *experiment2*, we use No. 33 tool of *netwox* to launch ARP cache poisoning to attack *nodew1* in *experiment1* as shown in Figure 4. The figure modifies that wed like to modify the MAC address of *nodew2* to $0C:0C:0C:0C:0C:0C$. Figure 4 also shows the table after the attack, which suggests that the modification succeeded.

Experiment Result and Analysis: We validate the result by checking the ARP table again as seen in Figure 5. The ARP table is successfully modified. Given this modified ARP table, any traffic sending to *nodew2*(to its IP address) will not be received by it due to the wrong MAC address. This results in a DOS attack at *nodew2*.

```
[lidawei@nodew1 src]$ sudo /usr/local/bin/netwox 33 -d ath0 -a 0C:0C:0C:0C:0C:0C
 -b 00:17:9A:C3:65:24 -c 2054 -e 2 -f 0C:0C:0C:0C:0C:0C -g "10.1.1.3" -h 00:17:9
A:C3:65:24 -i 10.1.1.2
Ethernet
| 0C:0C:0C:0C:0C:0C->00:17:9A:C3:65:24 type:0x0806
ARP Reply
| this answer : 0C:0C:0C:0C:0C:0C 10.1.1.3
| is for      : 00:17:9A:C3:65:24 10.1.1.2
```

Fig. 4. Launch ARP Cache Poisoning

```
[lidawei@nodew1 ~]$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
control-router.emulab.n  ether   00:B0:8E:84:69:34   C                     eth4
nodew2-lan0              ether   0C:0C:0C:0C:0C:0C   C                     ath0
```

Fig. 5. ARP Cache after Attack

2) Virtualization Issue: We present experiments on two subcategory for the virtualization issue, namely, *the virtualization bugs* and *resource exhaustion.*

a. Virtualization Bugs:

Experiment Description: In this experiment, three slices with slice names *test1*, *test2* and *test3* are created with the same topology of two Vnodes named *shared1* and *shared2*, and a link of bandwidth 100Mb/s connecting them. Tool Iperf is installed on both nodes. All the resources acquired are summarized in TABLE I.

TABLE I
ALLOCATED RESOURCES

Slice Name	Node Name	Host Name
test1	shared1	pc175.emulab.net
test1	shared2	pc172.emulab.net
test2	shared1	pc172.emulab.net
test2	shared2	pc175.emulab.net
test3	shared1	pc263.emulab.net
test3	shared2	pc102.emulab.net

First, only one Iperf server is running on slice *test1* at the node *shared1*. From the nodes *shared2* of both slices *test1* and *test2*, we try to connect to the server *shared1* with the following command: *iperf -c shared1*. Supposedly, each *shared2* will connect to *shared1* in its own slice respectively. Then we observed from the screen of the Iperf server (*shared1* at *test1*) that both of the clients connected to this same server even though they are not from the same slice, i.e. the nodes can communicate across slices!

In Figure 6, we illustrate our experiment with the screen captures of the four nodes. The left sides are the Iperf server and client in *test1* and the right side are those in *test2*. Figure 6 shows that the server *shared1* in *test1* (*slice1892*) (the upper left terminal) is connected by clients with ports in the sequence (numbers in the red circle): 43589, 53256, 53257, 43590. On the other hand, the first client *shared2* in *tests1* (*slice1892*, the left lower terminal) connected to the server with ports in the blue circle: 43589, 43590, etc. The second client *shared2* in *test2* (*slice1893*, the right lower terminal) connected to server with ports in green circle: 53256, 53257, etc.

However, it seems that the problem could due to the fact that the two slices share the same physical resources (*pc175* and *pc172*). So we performed the same experiment with *test1* and *test3* which do not share physical resources. We obtained the same result as shown in Figure 7. Further, we tried other possibilities including changing Vnode to a normal node, connecting to the Iperf server with IP address and using different node names for different slices (no matter whether it is a Vnode or a normal node).

Experiment Analysis: After the many tests, we found that cross-communication could happen only when the nodes are Vnodes with the same node name and Iperf to a server through the node name. This may be caused by the implementation

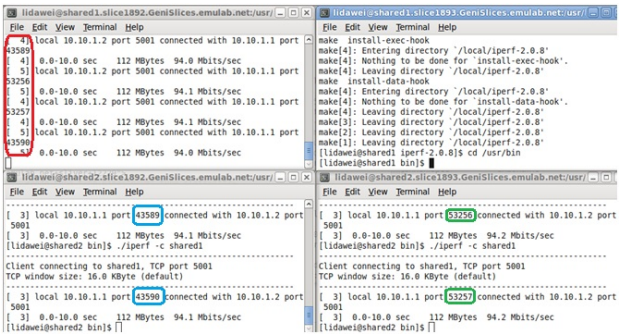

Fig. 6. Cross-slice Experiment (I)

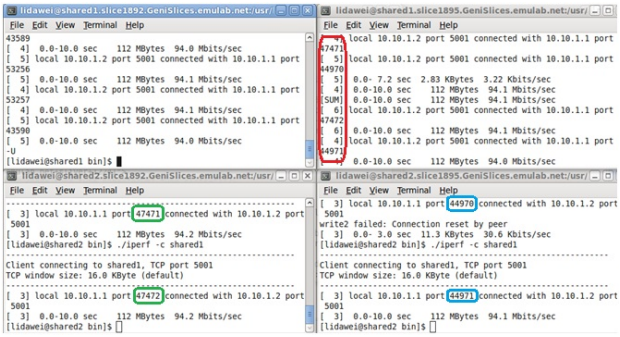

Fig. 7. Cross-slice Experiment (II)

of Vnodes in control framework and the mapping of the names. And the way for encapsulating the shared Vnode has a potential bug or drawback. So attackers can make use of this bug to send malicious traffic to a normal ProtoGENI experiment and harm the experiment result. Based on our experiment report, the developer team at University of Utah has fixed the issue described in this paper.

b. Resource Isolation:

The virtualization technology adopted by ProtoGENI should allocate a dedicated computing resource for a particular user which cannot be affected or exhausted by another user who acquires a virtual machine in the same physical machine. We test this function through the following experiment.

Two slices with slice names *vnode1*, a normal slice, and *vnode2*, the attacker's slice, are created. Each slice has a single Vnode located at the same physical node *PC263.emulab.net*. The attacker install the tool *stress* in his virtual machine to exhaust the computing resources. If the attack is successful, the computing resources such as CPU and memory of the normal slice *vnode1* will be reduced.

Experiment Steps: In the normal slice *vnode1*, we uses the Linux command *top* to show the CPU and memory usage before the attack (Figure 8). The figure shows that the CPU is free with 0.0

The attacker runs the *stress* software in his Vnode with the command:

$$stress\ -cpu\ 2\ -vm\ 1\ -vm-bytes\ 256M\ -timeout\ 30s$$

This command imposes a load by specifying two CPU-bound processes and one memory allocator process requir-

```
top - 15:46:07 up  1:12,  1 user,  load average: 0.00, 0.00, 0.00
Tasks:  13 total,   1 running,  12 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   2072116k total,   984884k used,  1087232k free,   138280k buffers
Swap:   128512k total,       48k used,   128464k free,   697040k cached
```

Fig. 8. Resource Usage before Attack

```
top - 15:50:34 up  1:16,  1 user,  load average: 0.00, 0.00, 0.00
Tasks:  13 total,   2 running,  11 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   2072116k total,  1238216k used,   833900k free,   139596k buffers
Swap:   128512k total,       48k used,   128464k free,   697084k cached
```

Fig. 9. Resource Usage after Attack

ing 256MB memory size. As the command is running, we observed changes of the resource usage in the normal slices Vnode (Figure 9). It shows that the CPU is still free meaning the processor is isolated and dedicated for a Vnode. But the memory usage is increased to 1238216KB, i.e. the memory is shared among multiple Vnodes.

Experiment Analysis: The above experiment shows that the virtualization technology used in ProtoGENI will allocate a dedicated CPU resource to a particular Vnode. But physical memory resource is not isolated among different Vnodes residing in the same physical machine. Attackers can use this drawback to exhaust the memory resource so that normal ProtoGENI experiments will be damaged.

C. Data Plane to Control Plane Attack

Data plane to control plane attack is the kind of attacks launched by a malicious ProtoGENI user to attack the ProtoGENI control network. Two attack experiments are performed in this subsection.

1) DOS Attack to Control Router:

Experiment Description: Since the ProtoGENI nodes are in the same LAN with a common control router, in this experiment, we use *netwox* to launch ARP cache poisoning attack to terminate the connection between the control router and an experiment node. As a result, the physical experiment node will not be available to other users who include this particular node in their resource request RSpec.

Two slices with slice named *experiment1* and *experiment2* are created. *Experiment1* is alive with a single node *node1* (physical node *pcwf146*) installed *netwox* for conducting attack. *Experiment2* is a pending slice for acquiring a specified node pcwf142 as its experimental resource. There are two ARP cache poisoning directions to achieve the DOS attack.

a. Poison the ARP cache of the control router:

We use the *netwox* tool No. 33 to poison control-router's ARP cache by spoofing a faked MAC address for pcwf142. The tool will send an ARP message to control-router. In order to prevent the expiration of the spoofed ARP entry, we use a shell script to run the *netwox* command repeatedly. We would like to modify the MAC address of the pcwf142 to $0C:0C:0C:0C:0C:0C$ in control-router's ARP cache. Then we try to acquire the *pcwf142* for *experiment2* to see if the resource is still available without any exception.

Experiment Result and Analysis:

We observed that the *experiment2* still acquired the *pcwf142* as its experiment node. This proofs that the control-router won't allow its ARP cache to be modified, which is also explained later by ProtoGENI developers.

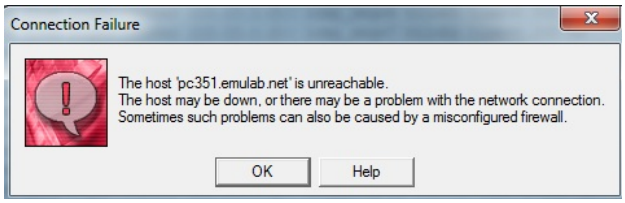

Fig. 10. Connection Failure

```
[lidawei@right ~]$ ping left
PING left-center (10.10.1.1) 56(84) bytes of data.
64 bytes from left-center (10.10.1.1): icmp_seq=1 ttl=64 time=0.242 ms
64 bytes from left-center (10.10.1.1): icmp_seq=2 ttl=64 time=0.155 ms
64 bytes from left-center (10.10.1.1): icmp_seq=3 ttl=64 time=0.192 ms
64 bytes from left-center (10.10.1.1): icmp_seq=4 ttl=64 time=0.228 ms
64 bytes from left-center (10.10.1.1): icmp_seq=5 ttl=64 time=0.265 ms
```

Fig. 11. VLAN is still Connected after Attack

b. Poisoning the ARP cache of the desired node:

We launch the ARP cache poisoning to attack the pcwf142 to modify the ARP entry about the control-router. Then, *experiment2* tries to get *pcwf142* as its resource node.

Experiment Result and Analysis:

This time, the result is inclined to the attacker, i.e., the ARP table at *pcwf142* has the wrong MAC address about the control-router. *pcwf142* is no longer available. We observed a series warning and error messages returned in the user's local interface which indicate that pcwf142 can not be used. The messages read like: " *pcwf142 appears wedged; it has been 6 minutes since it was rebooted.*", "$node\_reboot-reboot\_node$: *pcwf142 appears dead; will power cycle.*", "$ERROR : node\_reboot - reboot$: *Powercyle failed for pcwf142*" and "*pcwf142 may be down.*"

The experiment shows that the DOS attack will compromise the availability of ProtoGENI resources.

2) VLAN Issue:

Experiment Description: With VLAN, the issue is whether the virtual link connection in the data plane is isolated from the physical link in the control network. Or, say, will a failure of a physical link affects the virtual link? In this experiment, we use an experiment to answer this question. Again, two slices are created: slice *vlan* is a regular slice with two experiment nodes *left* and *right*; and slice *attack* is attacker's slice including a single node with *netwox* installed.

The attacker launchs ARP cache poisoning attack to disconnect the normal node *left* from the control-router using the method we described earlier. The result is that the local machine of the normal user cannot access the *left* via ssh (Figure 10). We then let the normal node *right* to *ping* the node *left* through the VLAN connection.

Experiment Result and Analysis: The result shows that the *left* is still reachable by *right* (Figure 11). The experiment result suggests that the VLAN in ProtoGENI is actually a LAN composed of experiment nodes with separate real network interfaces. The failure of the control network connection will not affect the experimental topology.

D. Data Plane to Internet Attack

The attack form Data Plane to Internet is the kind of attacks launched by a malicious ProtoGENI user to attack hosts or servers residing in the Internet. We perform two of attacks for example.

1) Sending unknown virus or worm to outside Internet: A ProtoGENI node can connect to a server or a normal host residing in Internet. This could give the potential to be abused by malicious users or hackers, who can send unknown viruses or worms to Internet. This kind of attack is fairly easy to launch in ProtoGENI as it allows an experiment node to upload files to a file server unaltered. A malware can thus be uploaded to or downloaded from a ProtoGENI node undetected. A proposed prevention approach for this kind of attack is to encrypt data retrieved from the ProtoGENI so that it cannot be executed automatically by the innocent nodes immediately [8].

2) Launching flooding attack: The ProtoGENI nodes enjoy high bandwidth connections with Internet. This benefit potentially helps to launch flood attack on a certain host or server. A very simple flooding attack can be *ping flood* where the attacker overwhelms the victim with *ping* packets if the attacker has a larger bandwidth than the victim. In this experiment, we try to see if the *ping flood* is possible. We also verify the connection of a ProtoGENI node to Internet and to test the traffic throughput. . We use a single ProtoGENI node and a PC from our lab. Both hosts have Iperf installed to test the connectivity and the bandwidth. The configuration is shown in Table II

TABLE II
RESOURCE ALLOCATION 3

Node Location	Operating System	Ethernet Card
ProtoGENI Node	Fedora 10	1000 Mbps
Lab PC	Ubuntu 10.04	100 Mbps

In our experiment, we set one machine as the iperf server and the other machine as client. In the client host, we connect the Iperf server by running:

$$iperf \ -c \ serverIP \ -t \ 30$$

The "t" flag is to set the test time to be 30 seconds. The collected data is listed in Table III (Throughput is the average over 40 experiments).

TABLE III
BANDWIDTH TEST RESULT

Iperf Server	Iperf Client	Throughput
Lab PC	ProtoGENI Node	91.7 Mbps
ProtoGENI Node	Lab PC	54.7 Mbps

Experiment Analysis: The Table III shows that the traffic sent from a ProtoGENI node to the lab node in Internet fulfills most of the lab node's bandwidth, which makes the lab node a victim (with lower bandwidth) of the flooding attack. In contrast, the throughput from the lab node to the ProtoGENI node is far less than the ProtoGENI nodes bandwidth, and it is even less than its own bandwidth of 100Mbps. The result suggests that the large bandwidth of a ProtoGENI node has the potential to overwhelm a lab node with low bandwidth in Internet.

E. Summary of Findings and Suggestions

We summarize our findings briefly and give suggestions for preventions. The results of our experiments show that the risk can be defined in two major aspects. First, it is possible to compromise the confidentiality and availability of ProtoGENI when other experimenters try to utilize the resources of victim nodes. Secondly, it is possible to use ProtoGENI resources as a potential source to launch attacks against the experimental machines at the Internet side.

(1) The control plane - data plane architecture leaves attackers (malicious ProtoGENI users) the chance to launch ARP cache poisoning attack and lead to denial-of-service to the control router. To prevent and detect these attacks, it is necessary to install ARPwatch or similar daemon tools on control routers and experimental nodes to detect and prevent malicious ARP packets.

(2) The wireless nodes available at the Utah site are located at a relatively small and concentrated area. This makes interference on a neighbor slice possible by the packet sniffing and spoofing. Network monitoring tools can be used to identify abnormal packets. For resource owners, more consideration should be paid on the node distribution.

(3) Current virtualization technology used by ProtoGENI Vnodes has had bugs leading to cross-slice traffic. Virtual machines residing on the same physical nodes share system memories which are likely to be exhausted by a malicious user. The resource providers and control framework development team should test more on the selected virtualization technology regarding to the isolation issue.

V. CONCLUSIONS

In this paper, we analyzed the security vulnerabilities of ProtoGENI We performed a series of experiments to verify potential vulnerabilities mimicking three different types of attacks: data plane to data plane attack, data plane to control plane attack and data plane to Internet attack. The results of the experiments indicated the potentials of breaching of confidentiality and availability internally within GENI, as well as the risks to external Internet. The prevention suggestions are given based on our findings.

REFERENCES

[1] Emulab Tutorial, http://www.protogeni.net/trac/emulab/wiki/Tutorial.
[2] GENI Global Environment for Network Innovations Spiral 2 Overview. http://groups.geni.net/geni/attachment/wiki/SpiralTwo/GENIS2Ovrvw060310.pdf.
[3] GENI Global Environment for Network Innovations Spiral 2 Security Plan, http://groups.geni.net/geni/wiki/SpiralTwoSecurityPlans.
[4] ProtoGENI wiki page, http://www.protogeni.net/trac/protogeni.
[5] X. Hong, F. Hu, Y. Xiao. GENI Spiral Two Project: GENI Experiments for Traffic Capture Capabilities and Security Requirement Analysis, http://groups.geni.net/geni/wiki/ExptsSecurityAnalysis.
[6] S. Peisert. GENI Spiral Two Project: The Hive Mind: Applying a Distributed Security Sensor Network to GENI , http://groups.geni.net/geni/wiki/HiveMind.
[7] GENI: Exploring Networks of the Future, http://www.geni.net.
[8] DETER testbed for security research. http://www.isi.deterlab.net/
[9] GENI Security Architecture http://groups.geni.net/geni/wiki/GENISecurity.
[10] Dawei Li, Xiaoyan Hong, "Practical Exploitation on System Vulnerability of ProtoGENI," *The 49th ACM Southeast Conference,* Kennesaw, Gegoria, March 24-26, 2011.
[11] Netwox http://ntwox.sourceforge.net/.