ExptsSec: S2.b. Report on initial experimentation on ProtoGENI and findings

Xiaoyan Hong, Fei Hu, Yang Xiao
Jingcheng Gao, Dawei Li, Dong Zhang,
University of Alabama
March 22, 2010

This document describes the experiments performed according to in the design document and the findings as the results of the experiments. The experiments on ProtoGENI/Emulab are performed following the three issues and targeted at the corresponding potential security vulnerabilities. Preliminary suggestions for GENI security requirements are presented in the summary section.

1. Authentication

1.1. Port Scanning

1.1.1. How does this works?

The ProtoGENI nodes are actual hosts on the Internet. Because Port Scanning is one of the most popular reconnaissance techniques attackers use to discover services they can possibly break into. It is natural for us to use a port scanning tool to detect what kind of vulnerabilities these nodes might have. After reviewing the scanning result, we can possibly find whether the hosts within the ProtoGENI are actually under certain security threat or not.

1.1.2. Experimental Design

First, we need to create a ProtoGENI experiment with several nodes. Then, we can get the nodes IP and using ports scanning tools to scan the host and store the results. Finally, we can analyze the result and see whether we can find any vulnerability.

1.1.3. Demo

First, we got the nodes from the ProtoGENI:

```
Permission denied (publickey,password,keyboard-interactive).
kofawp@kofawp-desktop:~/testbed/protogeni/test$ ssh kofawp@pc116.emulab.net
[kofawp@geni2 ~]$
```

Then, we transform it into IP addresses so that our tools can scan it.

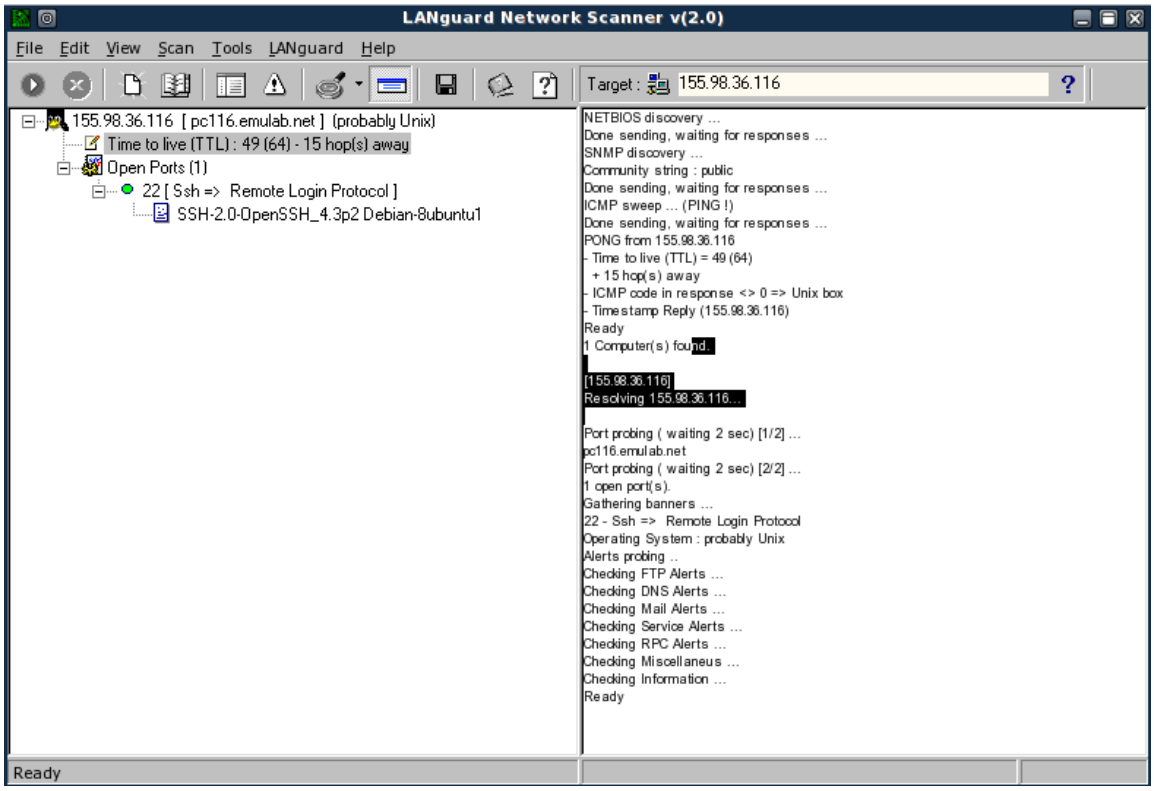
```
Tracing route to pc116.emulab.net [155.98.36.116]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  130.160.46.1
  2    <1 ms    <1 ms    <1 ms  UA-004-R1.ua.edu [130.160.4.1]
  3     5 ms     5 ms     5 ms  130.160.15.233
  4    12 ms     6 ms     5 ms  143.215.193.1
  5    22 ms    21 ms    21 ms  tge-0-1-0-1.chic.layer3.nlr.net [216.24.186.34]

  6    46 ms    47 ms    47 ms  denv-chic-36.layer3.nlr.net [216.24.186.5]
  7    59 ms    59 ms    59 ms  216.24.184.178
  8    59 ms    59 ms    59 ms  ebc-p-b-171.uen.net [140.197.252.87]
  9    59 ms    60 ms    61 ms  ebc-pe-a-178.uen.net [140.197.252.85]
 10    59 ms    59 ms    59 ms  205.124.249.118
 11    59 ms    59 ms    59 ms  199.104.93.33
 12     *        *        *     Request timed out.
 13    60 ms    59 ms    60 ms  155.98.127.45
 14    59 ms    59 ms    59 ms  155.98.127.46
 15    59 ms    60 ms    59 ms  pc116.emulab.net [155.98.36.116]

Trace complete.
```

Port Scanning
Result:



We only found one port 22 is open- SSH running on it, which is supposed to be. We still need further experiment to collect the data and see whether we can crack the SSH service on the ProtoGENI testbed.

This also indicates that most of the ProtoGENI machine's ports are closed (mean safe).

2. Experiment run-time interaction with ProtoGENI

When a user of ProtoGENI performs experiments, he needs to use the python scripts provided by the developer to use the APIs with XMLRPC over http and SSL (users could write their own scripts with a language he prefers as long as it supports XMPRPC). As is shown in the ProtoGENI tutorial, a user follows a series of steps to do experiments with the provided scripts. We analyzed these steps and also a few test scripts, and performed related experiments to explore potential vulnerability. One of the purposes is to find the weakness in handshake procedure (like TCP SYN attack). The details are described below with our findings.

2.1 Getting Ready Phase

2.1.1 SSL Certificate

The test script will look for the SSL certificate and pass-phrase in $HOME/.ssl/. For most of the users' convenience at this Spiral 2 phase, the code will save the certificate and pass-phrase in $HOME/.ssl/ other than using a command line argument.

Security issues:
The location makes it easier for being stolen or tampered with. If so, the attacker can obtain all the authorities.

Experiment Details:
User1 stole User2's SSL certificate and pass-phrase from /home/user2/.ssl and put them into /home/user1/.ssl (in the same local machine). User1 can create whatever slices he wants under User2's name. Further, if User1 can guess the slice name, say, "myslice", that the User2 has created, he can deleted the sliver of this slice , or this slice, or he can create another sliver within the slice.

2.1.2 SSH Keys

This is a step that could incur similar problems as to the SSL certificates. SSH keys are also saved in local machine with a well known location. Potential problems see the described in 1.1 on SSL certificates.

Security issues:
If stolen, attackers can access to the experimental nodes being used by legal users. From these nodes, more security attacks could be performed.

Experiment Details:
User1 can steal user2's SSH private key from /home/user2/.ssh directory. If user1 also gets user2's Emulab account name (not necessarily SSL certificate), he can try to poll pc**.emulab.net and log in to the nodes obtained in user2's experiment. User1 may have User1's ssl certificate as well using the method given previously. Many damages can be further introduced.

2.2 Using the Test Scripts Phase

2.2.1 test-common

Code Analysis:
For all the test scripts provided by the developer, they will all execute the test-common.py file first. This file defines the do_method which will be used by all other scripts to call the XMLRPC server over http and SSL. This file also helps to analysis the arguments in user's commands. The file is the core of all the scripts, so the attacker can just easily make change to the test-common.py such as adding a joking print to the code lines of printing "all the resources are busy, please try later". This will be an easy way to confuse the user.

Security issues: once changed can affect all the scripts.

2.2.2 Create Sliver

The scripts provide 2 ways to create the sliver: The first one will be using the createsliver.py file. This file uses a do_method("cm", "CreateSliver", params, version="2.0") as a very convenient way.

The second way to create the sliver is to use the two files getticket.py and redeemticket.py. We try to perform a SYN flood like attack as of only getticket but not redeemticket. The result shows that the system does not allow multiple getticket from one slice, so the server will not hang of DoS.

Security issues: possible DoS attack.

Experiment Details: user1 try to launch a DoS attack by implementing multiple getticket while not implementing redeemticket. However the server will return an error: Must release unredeemed ticket first: Could not get ticket".

Findings: the current ProtoGENI will not allow ticket-flooding DoS to happen.

3. Aggregate components and management

ProtoGENI resources for experiments and many system components relate to virtual machines and OsSs. Potential vulnerabilities in these systems will to security issues of ProtoGENI. Our first chosen OS is FreeBSD, currently Emulab uses it.

Experiment steps:

Step1: Set up the FreeBSD system (32bit version) on our virtual machine and log in with root account.

Step2: FreeBSD has been documented a security hole. That is, it allows other users to scan its port 23 and using such port for remote access – telnet. Huge amount of attacks on the FreeBSD have based on such method. We tried to using two popular software to test port 23.

- Software superscan3.00 to scan port 23: We selected a range of IP addresses used by a local lab (including the computer installed FreeBSD) and limited to port 23 only. From the returned information of the software, all we need is to find which IP address has an open port 23 and with the symbol "..%". The latter is the indication of a computer runs FreeBSD. Thus, this method allows us to find the IP address of a computer running FreeBSD and also to test whether the port is open or not. The result indicates that potential safety risk is at the same level of the vulnerability of FreeBSD.

- Software Fluxay 5.0: this software can scan port 23 and also try to get the host name if vulnerability exits. Our test run over our virtual machine failed in obtaining the host name. As such, with our preliminary trial on this particular issue, FreeBSD is safe.

4. Summary

In summary, some of the experiment findings showed safe cases, e.g., in terms of port scan, host OS (FreeBSD) identification, and ticket flood. They may only pertain to the experiments we performed. More investigations are needed to draw a firm conclusion. Vulnerabilities due to the locations of SSL certificates and PASSPHASE and SSH keys are subject to the accessibility to the local machines or

subject to the Torjan horses. However, we discovered that the real IP addresses of the experimental nodes are traceable from external machines, also there is a clear mapping relationship between the vnode names and their IP addresses. The former opens a pathway to external attacks, and the latter makes IP address scan easier. We'd like to suggest to have a way to hide real IP addresses and also to randomize the mapping between hostnames and their IP addresses.