Design and Implementation of Malware Dynamic Analyzer using Network Emulator GINE

M2011MM048 MITSUEDA Yasuaki
Supervisor : OKUMURA Yasuyuki

1 Introduction

Recently the number of malicious programs have been increasing in a short period of time. We need to know their behavior to cleanse them.

Although dynamic analysis is not able to figure out exact malware behavior, it is able to analyze malware behavior in a short period of time. It executes the binary in a disconnected Internet environment and analyzes malware behavior. However, there is a drawback with dynamic analysis. Most malwares try to connect to the Internet to connect to the corresponding server that malware's creator prepares.

To overcome this drawback, there are two previous works for emulating the network. The first one is "Improving Isolated Sandbox using Fake DNS Server"[1]. It use Isolated Sandbox using a number of computers. It provide the Fake DNS service with malware. The fake DNS service replies predefined domain name or IP address in case the domain name or IP address are not in the list.

Secondly, "Trumanbox"[2] uses two computers. One computer provides malware with servers such as HTTP, FTP, IRC, and SMTP. The other runs malware samples connected to one another. It redirects malware traffic to servers using ebtables and iptables command. This study analyzes the traffic to figure out malware behavior. Therefore, it redirects the traffic, only between servers and malware. Furthermore, it provides the DNS service which resolves the requested hostname to predefined IP addresses.

In this paper, we extend previous research[3] in 2012. We use the network emulator GINE[4] and the virtual machine QEMU to make a disconnected the Internet environment on one computer. We make malware connect to several virtual hosts and observe malware infection action. In addition, by providing fake servers, we figure out malware behavior.

2 System Overview

In this section, we explain virtual networks using QEMU and GINE.

2.1 Test Network Configuration

In this study, we make two virtual networks. The first one is the dissconected internet environment (see Fig.1).

- Virtual Windows Host
  We install virtual Windows OS on QEMU. QEMU is Open Source and it is possible to make a connection, using tap function, between guest OS and GINE nodes. Therefore we use QEMU as a virtual machine. By using the virtual machine, we are able to clean up the guest machine. We explain the method of connection between QEMU and GINE node in section 3.1.
- Virtual Switch
  Using kernel bridge, QEMU connects to the Internet directly. In this case, GINE nodes are impossible to connect to QEMU host. Therefore, we use virtual GINE switch, without assign IP addresses, which is possible to make a connection between QEMU host and GINE nodes.
- Virtual NAT Host
  We translate, using iptables, virtual Windows OS's destination IP address into fake server's IP address. We configure iptables rule below.
  "-d 0.0.0.0/0 -j DNAT -to-destination A.B.C.D"
- Virtual Fake Servers
  We provide malware with fake services such as HTTP, FTP, IRC, SMTP and DNS.

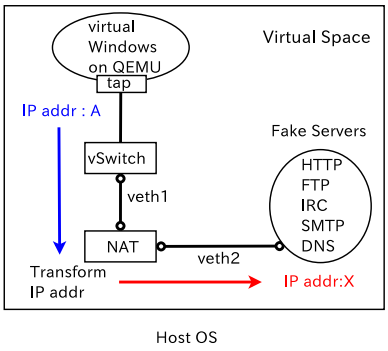

Figure 1 System Architecture

Providing malware with fake servers, we always have to assign IP address. Malware always tries to connect to the original server. In this case, malware connects to fake servers. For example (see Fig.1), virtual Windows host connects a DNS server's IP address "A". Virtual NAT host receives malware's packets on veth1 and translates malware's destination IP address into the fake server's IP address "X". In addition, it forward the packets to the fake servers.

Secondly, we make a connection between malware and the original server (see Fig.2).

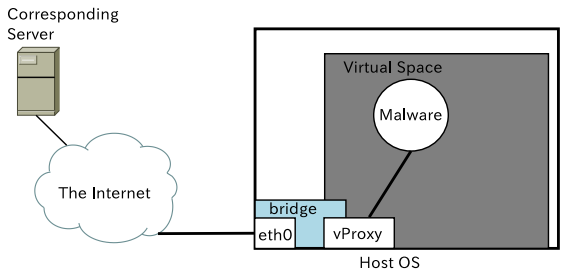

Figure 2 The transparent proxy

"vProxy" is a virtual GINE host and the transparend proxy host. It only permits port 80(HTTP) and port 53(DNS) using iptables command to prevent the infection. We make malware connect to the original server using network bridge. The bridge can make a connection between NIC(eth0) and virtual NAT host's NIC.

Therefore, we can make malware connect to the original server with benign.

2.2 Network and Hosts Virtuarization

GINE(Goto's IP Network Emulator) emulates IPv4/v6 network which component some router and link. GINE uses NETNS(Network Name Space) which is a Linux kernel virtual host. It uses only network virtualization and do not use file system nor process virtualization since it is rather inconvenient for network emulation purpose. Since network interfaces including loopback interface(lo) and Unix domain sockets are not shared among virtual network stacks and the host OS 's original network stack, a special virtual network device, called Virtual Ethernet Pair(vethdevice), must be used.

GINE can not emulate Windows hosts. Therefore, we use QEMU to make networks including Windows hosts and NETNS hosts. QEMU has a tap function to connect to other virtual hosts.

3 Implementation

In this section, we explain the implementation.

3.1 Connection of Virtual Hosts and Routers

In this section, we explain the method to connect a virtual QEMU host and internal GINE hosts[3]. In the case of executing QEMU with default (user mode network), QEMU prepares NAT DHCP server, DNS server and Samba server for guest OSs. Consequently, other hosts are not able to connect to guest OSs since there is the NAT DHCP server for guest OSs. Furthermore, the guest OS can not connect to other guest OSs and the host OS. Therefore, we use QEMU's tap function. Guest OSs are able to connect other hosts by tap function. To use tap function, we have to execute QEMU with the options below.

- qemu -net nic, vlan=1 -net tap, vlan=1, ifname=tap0

"-net nic" is an option which creates virtual network card and "-net tap" connects tap device to QEMU. We use these options to make the connection between QEMU host and GINE host (see Fig.3).

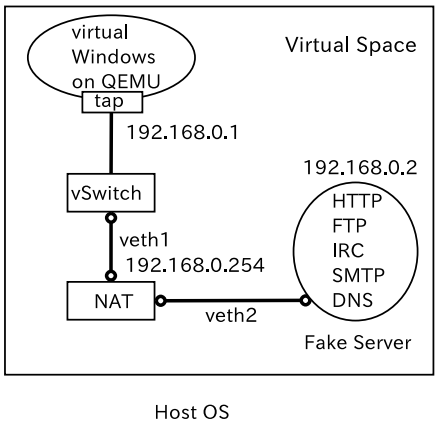

Figure 3 Connection of QEMU host and GINE host

GINE uses NETNS and veth to make a connection GINE hosts each other. Therefore, GINE hosts can not make a connection to tap device. Then, we connect veth1 to a virtual GINE switch which connect tap device. Therefore, we can make the connection between GINE host and QEMU virtual host.

However, some malware detect the virtual environment and stop execution itself to prevent analysis. In this case, we execute malware on a real computer. We use two real computers (see Fig.4).

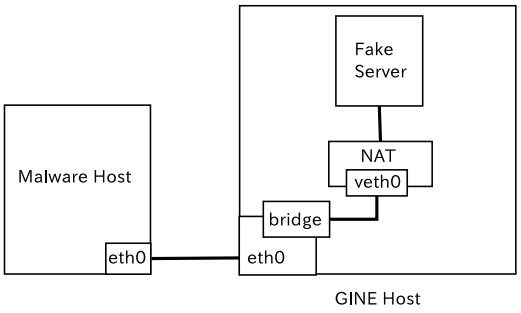

Figure 4 Using two real computers

One computer is a malware computer. The other one is a GINE computer which provides fake servers with malware. We connect the malware computer to the GINE computer with a crossing cable. The virtual NAT host connects to the malware host by bridging. Fake servers are the same as the previous section's fake servers.

3.2 Fake Servers

SMTP and DNS are complicate mechanisms. Thus, we use standard server software. For SMTP, we use Postfix. In addition, we use NSD and Unbound for DNS.

- Fake DNS server
  We use NSD for fake DNS root server and set up fake DNS server on the virtual Internet and configure zone file (see below). Fake DNS server always replies the same IPv4 address for A record query by using wild-card. For AAAA query, it replies with the same way. For reverse query, we implement a program which always replies the same domain using unbound library. However, we do not implement this since it is low-priority[5].

  zone file configuration(IPv4)

  ```
  * IN A 1.2.3.4
  ```

  We configured the zone file and set fake DNS server's IP address(Ex:1.2.3.4) on the IP field. In this case, the DNS server always replies same IPv4 address(1.2.3.4).
- Fake SMTP server
  We use Postfix for fake SMTP server. We set up this server on the virtual Internet. This server only replies SMTP messages. it does not send E-mail for real.

On the contrary, HTTP, FTP and IRC services are simple mechanisms. Therefore, We use xinetd. xinetd is the Internet services daemon. Each time accepting connections, xinetd starts services according to port number. In addition, xinetd is able to redirect access to another port. In the case of starting services, xinetd can run server program and an arbitrary program. Hence, we implement server program easily.

We implemented the fake HTTP server program. This program always respond with same reply to malware.

This program reply these message below.

Messages of HTTP

```
HTTP/1.1 200 OK
Content-type: text/html

<html>
<head></head>
<body>
<h1>Hello, world!</h1>
<?php
print ("fake script");
?>
</body>
</html>
```

"200" is a HTTP message. It shows acceptance malware requirements. The fake HTTP sever always reply this messages.

We extended the ftpd[6] which is a opensource software. In the case of using ID and password, the fake FTP server ignores login ID and password which are sent. Therefore, malware always succeeds entering the FTP server, as if malware used the correct ID and password. In addition, the fake FTP server accept anonymous login. Every time malware requires files, the fake FTP server replies fake files according to file extension(exe, png, jpeg, jpg, gif, html, php and pdf). For example, malware requires A.exe file. In this case, the fake FTP server replies fake.exe file. In addition, the fake FTP server reply image files because malware requires image files which embedded malicious code.

We identify protocol by well known port. However, in some malware, well knownport numbers are not used. Identification of protocol is important for our study. Therefore, we identify protocol by payload protocol identification[2]. We identify protocol according to table 1. For example, There is "GET" in client's payload.

Table 1 Protocol Identification

Protocol	Pattern (at the beginnig)	Pattern (somewhere)
HTTP	"GET /"	
IRC	"NICK "	
FTP	"220"	"ftp" (case insensitive)
SMTP	"220"	"mail" OR "smtp" (case insensitive)

We identify HTTP. If there is no payload, we identify protocol according to destination port number.

3.3 Transparent Proxy with Body Removal

In some malware, our fake server can be detected. Moreover, if malware can connect to the original server, we can figureout exact behavior. Hence, we use transparent HTTP proxy. It forwards the original server packets to malware. However, we should not forward the original packets without any change. Therefore, we implemented the transparent proxy with perl since perl can manipulate text easily.

Client(malware) send http request to the original server. When forwarding the request, we have to use iptables command to redirect the packet to the transparent proxy. Then, the proxy establish the connection with non blocking mode to make wait the connection. In addition, it gets the client's destination IP address using "getsockopt" function. It sends the request to the original server by proxy. After receiving the HTTP reply, if the reply is 200s, it removes the HTTP body and forwards the reply to the client (see below).

Original HTTP headers

```
HTTP/1.1 200 OK
Date: Tue, 11 Dec 2012 10:03:53 GMT
Server: Apache/2.2.14 (Ubuntu)
Last-Modified: Thu, 29 Nov 2012 06:45:40 GMT
ETag: "7e22f2-10d-4cf9ca1aa15c0"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 205
Content-Type: text/html

Malicious Code Message Body
```

Malware gets a file by using HTTP get method. In the case of HTTP response message being 200s, the proxy replaces original HTTP body with benign body(see below).

Replaced HTTP headers

```
HTTP/1.1 200 OK
Date: Tue, 11 Dec 2012 09:35:35 GMT
Server: Apache/2.2.14 (Ubuntu)
Last-Modified: Thu, 29 Nov 2012 06:45:40 GMT
ETag: "7e22f2-10d-4cf9ca1aa15c0"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Length: 31
Content-Type: text/html

<html><body>HELLO!!</body></html>
```

The proxy removes HTTP body into "HELLO!!". The proxy does not do anything in the case of such as 300s, 400s and 500s. There is room to consider other protocol such as FTP.

4 Experimental Results

We analyze malware samples from MWS(anti Malware engineering WorkShop) 2012 Datasets[7]. These sample are collected between 1st and 31th Jan. in 2012 using 12 honeypots. In addition, We analyze samples using fake servers in next section.

4.1 Malware Analysis with Fake Servers

We classified these 10538 samples using clamAV. There were 17 classes and 11 samples which clamAV could not detect malware. We analyzed 1 sample for each class and the 11 samples (They are described in the top 5 digit hash). Therefore, we analyzed 28 samples. We executed 28 samples on the virtual windows XP for 10 minutes. 11 of 28 samples did not run on virtual Windows XP, virtual Windows 7(32bit,64bit) and Windows 7(64bit) on real a machine. We show the results of malware samples behavior (see Table.2). Then, we captured malware traffic. We describe a part of malware behavior.

Allaple sent ICMP request to a host(219.151.X.X) twice. Then, this sample tried to connect to the host with port 139(NETBIOS−SSN). We guess that it is a infection action.

Trojan.Inject, Trojan.Jorik, Trojan.Injector, Trojan .Kazy, 42b2f, 8e915, ce891, 56404 and e0428 showed

Table 2 Malware Classification and Behavior

Class	Used protocol or port number
Allaple	ICMP, netbios-ssn(port 139)
Trojan.Inject	DNS
Trojan.Crypt	DNS, microsoft-ds(port 445), port 555
Trojan.Dropper.Agent	microsoft-ds(445), DNS, port 976
Trojan.IRCBot	DNS, microsoft-ds(port 445), port 555
Trojan.Injector	DNS
Trojan.Jorik	DNS
Trojan.Kazy	DNS
W32.Virut	DNS, HTTP, port 555
42b2f...	DNS
8e915...	DNS
e95f7...	HTTP, port 7007
54932...	netbios-dgm(port 138)
ce891...	DNS
56404...	DNS
8746e...	netbios-dgm(port 138)
e0428...	DNS

same behavior. These samples sent same DNS request of TXT record. Therefore, we configured fake DNS server to reply TXT record and analyzed Trojan.Injector again. This sample only continued TXT record request. We guess that the action is DoS attack which using DNS query.

Trojan.IRCBot, Trojan.Dropper.Agent, W32.Virut, and Trojan.Crypt showed similar behavior. These samples tried to enter the IRC server with password. Trojan.IRCBot, W32.Virut and Trojan.Crypt used port 555. Trojan.Dropper.Agent used port 976. Trojan.IRC Bot sent "Password h4xg4ng", "NICK [00|JPN|XP|339 952]" and "User ID SP0-805 * 0 :M11MM0482LBFCJ" to the server. The others sent similar information. We guess that the password is used to enter the server and NICK is infected host information. In addition, USER ID was infected Windows user name. We guess that malware's creator collects infected host information to control. Furthermore, these samples sent ARP request to the same segment network hosts. Receiving ARP reply from a host, these sample tried to connect to the host using microsoft-ds. After these samples sending ARP request X.X.X.254, these samples sent ARP request to the next segment. We guess that this is the infection action.

e95f7 resolved a domain "www.google.com" and tried to connect to the domain. Then, it resolved a domain "www.what..." and tried to connect to the domain. This domain was a website which showed client's IP address. In addition, this website showed client's IP address as a image. This sample sent HTTP request "GET / HTTP/1.0" to the website. However, the website only replied for "HTTP/1.1" request. We guess that the website changed the specification. In addition, the sample sent "USER M11MM048-2LBFCJ n n :nservice" and "NICK P|000|JPN|XP-SP0|zkpregfscf|" to a domain "frayednd..." with port 7007. Furthermore, it sent ARP request to same segment hosts which is similar to Trojan.IRCBot. We guess that this is the infection action.

54932 and 8746e show same behavior. These samples continued sending packets with ports 137 and 138 to 192.168.0.255. We guess that this is the infection action.

After analysis, we tried to connect to each corresponding servers with the same port. However, we could not connect to servers.

4.2 Malware Analysus with TransParent Proxy

We did not analyze malware samples with transparent proxy since we could not connect to corresponding servers. Hence, we tested the proxy using the wget command. A host tried to get fake.exe file, using wget with transparent proxy, from our server. The host executed "wget h303.../fake.exe" command. Then, the host got the fake.exe. We show the fake.exe below.

Content of fake.exe

```
<html><body> HELLO!! </body> </html>
```

Although the host requested exe file, the proxy removed http body.

5 Conclusion

We made the disconnected internet environment on one computer using GINE and QEMU. Providing fake DNS, HTTP, IRC and FTP servers, we could figure out malaware behavior. This study is effective for types of malware that sends infected host information. There are many possible improvements. We show the improvements below.

- Spam analysis mail attachmented file
- Malware analysis with transparent HTTP proxy
- Implementation of transparent proxy with other protocols

Acknowledgments. This study was supported by the anti-Malware engineering WorkShop(MWS).

References

[1] S. Miwa, D. Miyamoto, H. Hazeyama, D. Inoue, and Y. Kadobayashi, "Improving isolated sandbox using fake dns server," in *Cyber Clean Center • Information Processing Society of Japan, anti Malware engineering WorkShop 2008 (MWS2008), http://www.iwsec.org/mws/2008/manuscript/ 1019.pdf*, Oct. 2008 (In Japanese with English summary).

[2] G. Christian, F. C. Freiling, M. Kuhrer, and T. Holz, "Truman box: Improving dynamic malware analysis by emulating the internet," in *13th International Symposium, SSS 2011, Grenoble, France*, Oct. 2011, pp. 208–222.

[3] M. Yasuaki and K. Goto, "Design and implementation of malware analysis using network emulator GINE," in *Computer Security Symposium 2012 (CSS2012), Vol. 2012, No. 3*, Oct. 2012 (In Japanese with English summary), pp. 114–121.

[4] K. Goto, "Network emulator with virtual host and packet diversion," in *Cyber Journals: Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Telecommunications (JSAT), Vol. 3, No. 3*, 2012, pp. 13–20.

[5] J. A. Morales, A. Al-Bataineh, S. Xu, and R. Sandhu, "Analyzing and exploiting network behaviors of malware," in *Security and Privacy in Communication Networks Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Vol. 50*, 2010, pp. 20–34.

[6] VA Linux Systems, "ftpd," accessed Feb. 2013, http://jftp.sourceforge.net/.

[7] anti-Malware engineering WorkShop, "About mws," accessed Feb. 2013, http://www.iwsec.org/mws/2009/en_about.html.