Global Information Assurance Certification Paper

An Exercise in Incident Handling

Advanced Incident Handling and Hacker Exploits
Practical Examination for GCIH

By:
Denis Calderone
February 17, 2001

Executive Summary

This report will outline an incident that occurred to a client of mine that involved a breach in a Linux system that existed outside their DMZ, on an exterior network segment. I will refer to this company as the XYZ Widget Company for the purposes of this report. As this company's security consultant, I was engaged to coordinate the incident handling effort. This report will outline how we brought an incident team into play, and how we contained and eradicated the security breach. I will also go into some detail on architectural redesign work that was proposed for their network following the incident.

The site in question was constructed to function as an e-commerce site, and was in the business of collecting credit card information and fulfilling order requests. With this fact in mind, the company's executives made it top priority to assess the level of the security breach and to take quick and decisive action based on the findings of the investigation.

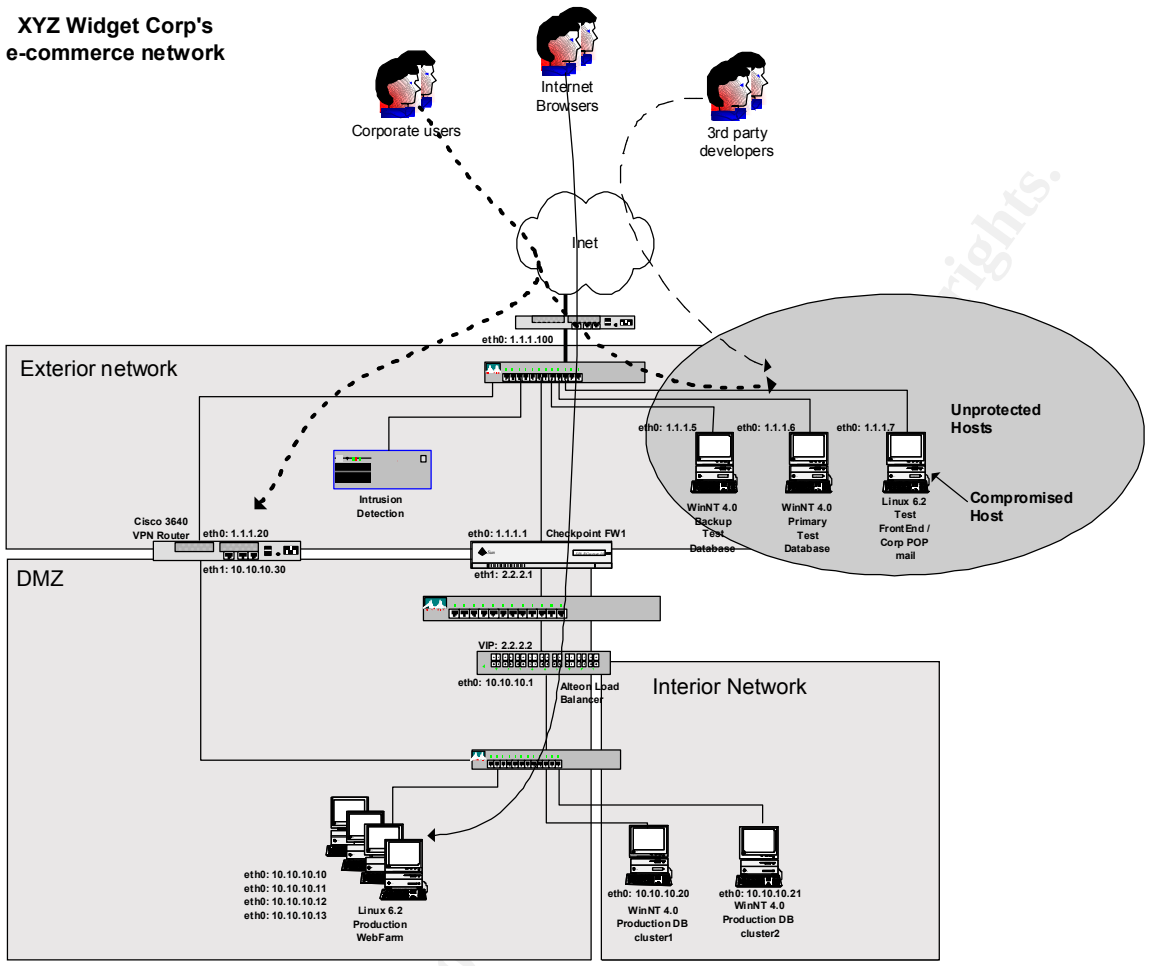
The site is made up of three zones: an external network where the firewall, vpn concentrator, IDS, and test systems all reside; a DMZ where the outside interface of an Alteon load balancer sits; and an internal network consisting of 4 Linux web servers, and 2 clustered MS SQL machines running on NT 4.0. All machines on the interior network were using RFC 1918 addresses. There are three distinct user communities: clients and web browsers, that will shop on the site; corporate users that will access the site through the vpn concentrator; and the 3rd party developers that will access only the test systems sitting on the External network.

You can see, from the diagram below, that the DMZ houses only the virtual IP that the shoppers reference when accessing the site. Early plans for this site had the test equipment on a separate interior network segment stemming from the Alteon switch, but due to performance issues, brought to light by the 3rd party developers, who reside in the Ukraine, the administrators decided to place these machines in a vulnerable state.

There are three IP segments in the network environment.

- 1.1.1.0 – This is the Exterior segment. It houses the inside interface of the exterior router, the outside interface of the firewall, the outside interface of the VPN concentrator, the intrusion detection system, and the three test systems.
- 2.2.2.0 – This is the DMZ segment. It houses the inside interface of the firewall, and the outside interface of the Alteon load balancer. This is where the VIP resides for incoming requests to the web farm.
- 10.10.10.0 – This is the Interior segment. It house all 6 production servers. The web farm is technically part of the DMZ, since it is directly accessible from the Internet through the use of the VIP on the Alteon switch.

This report will show how this lax security posture allowed an attacker to breach my clients web site and to go undetected for several months, and how some very simple architectural and administrative changes were later implemented to better my clients overall security stance.



Preparation

The breach in security was found quite by accident after the IDS box was reconfigured to send alerts directly to an administrator whenever a high severity alert went off. An arbitrary buffer overflow attempt caused a small panic within XYZ, and my team was asked to investigate. After a quick port scan showed the existence of what was described by the responding engineer as a "strange" port listening on the Linux box, we quickly jumped into action.

1. XYZ's CIO was made executive of the operation. We would include him on every step of the investigation, and he would have final say on what was to be done.
2. We instructed XYZ to work with us, not against us, and not to touch anything without first discussing it with the team.
3. We instructed XYZ's CIO to inform their developers not to access any systems throughout the duration of the investigation.
4. An incident response specialist was brought in to perform the actual engineering behind the investigation.
5. An XYZ administrator was assigned as the on site technician that would perform any physical work that was deemed necessary.

6. And I played the technical lead coordinating the efforts between all parties, and providing post-consulting support to XYZ's CIO.

We decided to meet each morning at 10:00am for status reports, to keep everyone in the loop, and once more each afternoon to follow up on the day's activities.

XYZ Widget Corp. was very receptive to helping us, and allowing my team to do their work, we found no hard failings between XYZ's administrative staff, and we didn't get the feeling that anyone was trying to hide anything. The network configuration I was working with was much different than the secure network I had originally designed for them. I believe they felt humbled once it became clear that they were breached, and likewise I carefully instructed my team not to throw any "we warned you, should have listened" in their faces. We were determined to work together as a successful team, and help our client make the best decision possible.

Identification

As mentioned earlier, the breach was discovered after a reconfiguration of XYZ's IDS box caused an alarm to trip and page an XYZ administrator. The alarm warned of an attempted buffer overflow on one of the test systems. XYZ asked us if this was anything to be concerned with, and we told them that we would do a quick scan of the targeted box to determine if it looked like anything was out of place. The system in question was a Linux box, which served a duel function as both the company's POP mail server, as well as the test web server for the e-commerce project. This server resided at 1.1.1.7. The results of the scan are below:

```
$ nmap -sT -pi-65535 [redacted]

Starting nmap V. 2.12 by Fyodor (fyo@oswchp.com, www.insecure.org/nmap/)
Interesting ports on [redacted]
Port       State       Protocol  Service
21         open        tcp       ftp
22         open        tcp       ssh
23         open        tcp       telnet
25         open        tcp       smtp
79         open        tcp       finger
80         open        tcp       http
110        open        tcp       pop-3
111        open        tcp       sunrpc
113        open        tcp       auth
443        open        tcp       https
514        open        tcp       shell
515        open        tcp       printer
938        open        tcp       unknown
1024       open        tcp       unknown
3200       open        tcp       ppp
47017      open        tcp       unknown     (0 unhandled)
```

The port scan uncovered that the system was not hardened, and that there was something listening on port 47017. We tried connecting to this port and found that it looked to be responding with SSH commands. We found this immediately suspicious and assumed the worst. We consulted with the administrator's at XYZ, and confirmed that they had not setup any SSH daemons to talk on that port.

A quick search on securityfocus.com found that the T0rn root kit uses this port. Now we new that we had a real incident on our hands and we put the rest of the team into play.

I remained the primary contact during the course of the ensuing investigation, but I relied heavily on the Incident Response Engineer who originally got the call, and performed the initial investigation.

Upon contacting XYZ's CIO, it was determined that top priority was to determine if the production system was jeopardized at anytime during the breach.

Containment

Since the incident response specialist was some distance from XYZ Widget Corp's data center, it was decided to have him do his work remotely. I worked onsite with XYZ's network administrators. I brought copies of Linux and Windows NT 4.0, a CD containing all service packs and hot fixes, as well as a directory of clean Linux binaries, a laptop with configured with Win2K and running a copy of Linux 6.2 on a Vmware partition, installations of Nessus (vulnerability assessment tool), Nmap (port scanner), and Etherpeak (sniffer).

Our original recommendation was to take the infected system off-line, and perform our analysis at our forensics lab, but this was ruled out because of the need to keep the corporate mail system operational. The companies POP mail was served out of this test web server.

Our first line of business, before performing any investigative work, was to get a good backup of the Linux system. We got approval to shut down the Linux system for a short time. We used a Logicube Echo disk-cloning device to get a sector-by-sector copy of the Linux disk device.

INVESTIGATION OF TROJANIZED LINUX SERVER

Our investigation of the trojanized Linux box began with testing the state of certain elementary system components such as ls, ps, and netstat. As we suspected, running netstat did not show the system listening on port 47017. Additionally, running ps did not show any unknown processes.

We uploaded clean binaries to the system, and began investigating system changes. As there was no checksum taken of this system, we were flying somewhat blind. We started by looking for all hidden files and directories:

```
# for x in `ls -laR | awk '{ print $9 }' | grep "^\."`; do find .
-name $x; done | sort | uniq
```

We found a hidden directory /usr/src/.puta. In this directory we found a file that contained output from a sniffer dating back some three months. We now realized that

this intruder had been hanging around for quite sometime. The modification date on the sniffer output file showed that the last change was about 1 and half months after it was originally created.

Another hidden directory was found /usr/t0rn/info that contained files that had been modified since the sniffer had been shut off. This proved to us that the intruder was still accessing the site from time to time.

A full system search showed that a number of system files were changed at 1:30am on the morning that the packet sniff file was created, and we concluded from this, that it was probably at this time that the first breach occurred. Regrettably, this was prior to the installation of the IDS system and there were no logs recording this intrusion.

It appeared that the intruders went through some effort to cover their tracks. We could only find evidence of trusted users accessing the system through FTP, HTTP, and TELNET during the time period we assumed the intrusion took place.

INVESTIGATION OF EXTERIOR NT 4.0 SERVERS

Sitting on the same segment as the trojanized Linux box were two NT 4.0 servers that were serving as the test database environment. We wanted to determine whether there were any signs of trojanized code on ether of these two NT systems. Both systems had unnecessary services running, and were fairly open to attack.

Before performing our investigation, we copied the %systemroot% and all subdirectories, system files within c:\, and all MSSQL files and directories to the opposite NT machine for safekeeping. So the cluster1 SQL test box contained a backup of cluster2 SQL test box, and vice versa.

We began our investigation of these systems by testing to see what ports netstat showed listening on the device. Netstat did not show any unusual results though. We than performed a search for all files created or modified around the time of the initial attack on the Linux box, but there did not seem to be anything out of the ordinary. We took a close look at the ftp logs around the time of the initial attack on the Linux box. There were signs of unsuccessful login attempts, but nothing more.

We performed the same steps on the other NT system, but had similar results; we didn't see any sign of tampering, or break in on either test database server.

We concluded at this time, that it was unlikely that our attacker would have hacked the NT boxes, since he already had a sniffer on the Linux box, and was perfectly capable of stealing passwords in this fashion. We could have performed a more though search of accessed files on the NT system, but decided to move on with our analysis of how susceptible the systems sitting on the DMZ were to attack from the Linux box.

INVESTIGATION OF PRODUCTION SERVERS SUCEPTIBILITY TO ATTACK FROM EXT. NETWORK

To test the susceptibility of the production machines to an attack from the Linux server, we first tried simply to ping one of the machines at a 10.10.10.x address. This failed, so we tracerouted the address, and found that the failure was occurring at the first hop, 1.1.1.100, this is the address of the exterior router, and is also the default gateway of the Linux server, and the NT servers on the exterior network. We tried adding a default route to our test system, and pointing it to the firewall instead, but this did not allow us access to any 10.10.10.xx machines. We examined the firewall policy, and the routing table of the system where it resided, and found that there were no routes to the 10.10.10.xx network. We now concluded that there was no way to directly access the 10.10.10.xx systems from the Linux server or either of the two NT servers on the exterior network, but what about the VIP for the production web farm?

We tried to ping the VIP, 2.2.2.2, and the ping was successful. A traceroute to 2.2.2.2 showed that the packets traveled first to the exterior router, than to the firewall, and than to VIP on the other side of the router. Now we took a closer look at the firewall policy. ICMP was being allowed between any host, so that explained why the ping worked. There was a rule that allowed from source any to host 2.2.2.2 on ports 80, and 443. This was the only rule that allowed access between our exterior network, and the devices on the DMZ. Additionally, the firewall was configured to reject any traffic directed to any of its interfaces, from any directly attached network, so we felt pretty good that the firewall had not been compromised.

So we determined that any attack that the attacker could have posed on the DMZ would have had to be through the use of http, or https protocols. This limited the scope of attack points that our attacker could have used, but did not completely eliminate the threat, so our next step was to take a closer look at the machines sitting on the DMZ and on the Interior network segment.

INVESTIGATION OF PRODUCTION LINUX SERVERS

The work on the DMZ and Interior network machines was performed via a VPN connection through the Cisco VPN router that XYZ Widget Corp setup for us.

For each of the four Linux machines on the DMZ we:

- Ran a full nmap port scan
  - *nmap –sT –p1-65535*
- Ran a Nessus vulnerability scan
  - Selected all but dangerous modules
- Ran ps and looked for any unknown processes
  - *ps -ef*
- Compared the results of the nmap with the results of a Netstat
  - *netstat –a*
  - Visual comparison of results
- Uploaded clean binaries
  - *cp –f /mnt/cdrom/binaries/find /usr/bin/find*
  - *cp –f /mnt/cdrom/binaries/ls /bin/ls*
  - *cp –f /mnt/cdrom/binaries/ps /bin/ps*

    - *cp –f /mnt/cdrom/binaries/kill /bin/kill*
    - *cp –f /mnt/cdrom/binaries/vi /bin/vi*
- Rechecked open ports with Netstat and looked for deltas between the first and second run
    - *netstat –a*
    - Visual comparison of results
- Reran ps command and looked for deltas between first and second run
    - *ps -ef*
- Checked for open files with lsof
    - *lsof*
- Checked for file modified less than 90 days ago
    - *Find / -mtime –90 - print*

In all cases, this provided negative results. There did not appear to be any breaches of security on any of the secured production Linux machines. There was however many vulnerabilities, and unnecessarily open ports on each of the boxes. Additionally, logging was not enables on these machines.

INVESTIGATION OF PRODUCTION NT 4.0 SERVERS

For each of the two NT 4.0 machines on the Interior network, we:

- Ran a full nmap port scan
    - *nmap –sT –p1-65535*
- Ran a Nessus vulnerability scan
    - Selected all but dangerous modules
- Checked for unknown processes using Task Manager
    - Start/Run: taskmgr
- Compared the results of the nmap with the results of Netstat
    - netstat –n –a
    - Visual comparison of results
- Checked for files created or modified around the time of the original breach of the Linux machine on the exterior network segment
    - Start/Find/Find Files and Folders
    - Searched for all files within required date range
- Checked for files created or modified between 30 and 90 days ago
    - Start/Find/Find Files and Folders
    - Searched for all files within required date range

Again, our investigation of these production machines showed no sign of a breach.

Prior to finishing our investigation of the DMZ and Interior systems, we instructed XYZ Widget Corp. to change all system and user account passwords behind us, as well as deleting the VPN account we used to access the DMZ and Interior systems.

Eradication

Our conclusion was that it looked like the breach had occurred though a common exploit, probably through wu_ftp or rstatd. Our investigation showed a definite breach in the test Linux server, but no concrete evidence of unauthorized access was found on any of the other systems. It is our belief that nothing on any of the three externally available systems should be trusted and that those three systems should be removed and completely rebuilt with clean version of code.

We informed XYZ Widget Corp. that it seemed unlikely that the attack actually accessed any of the systems on the DMZ or Internal network segments. We explained that it was impossible to say for certain that these machines were untouched, but without an extensive forensics initiative, it would be impossible to say for certain.

At this point, the decision was made by XYZ Widget Corp's CIO to remove the exterior systems from the network. We were instructed to rebuild these systems using industry best practice in tightening the systems as much as possible.

We made several changes to the firewall to include no ICMP to the backend network, and an explicit rule stating that from source 1.1.1.0 to any on any port 'Reject'. This rule will make it impossible to stage any attacks between the exterior network to the DMZ or Interior networks.

Additionally, we convinced XYZ Widget Corp's CIO to demand his 3rd party developers begin using SSH to access the test environment instead of FTP.

To compliment the scans we had already run on the DMZ, we decided to run an external scan from the point of view of an attacker on the Internet. We looked for, and identified all weaknesses on TCP80 and TCP443 into the DMZ. To accomplish this task we needed to create a hard connection through the Alteon load balancer to one specific production server. We also checked the IDS logs during our scanning effort to make sure it picked up all critical attacks sent to the DMZ. All patchable vulnerabilities we found during the scans were patched during the recovery phase.

XYZ Widget Corp's 3rd party developers were alerted of the situation and asked to find the most current "trusted" backups. We implored with them the be extremely upfront about any code that may have been copied from the test systems into their development environment. We received mixed results with this line of questioning, and we walked away worried that the "trusted" backups, may not have been so trusted after all.

Recovery

We convinced XYZ Widget Corp, that if they were going to place systems outside the defenses, than these systems would need to become bastions, and strict guidelines would need to be followed.

We rebuilt the Linux system starting with a completely fresh format of the system drives. We were careful to follow good hardening guidelines while performing this rebuild. We determined that the only needed services, required for remote operation were TCP ports 80, 443, 110 and 22. Inetd was shut down, and we installed ipchains to control access to system. Most steps outlined in *the SANS Step-by-Step guide for Securing Linux* were followed.

The test Win NT 4.0 systems were also rebuilt. These systems were also reformatted prior to having most steps outlined in the SANS guide for Securing NT 4.0 applied. This time we installed WinRoute to tighten down access to these machines. It was decided that the only access allowed to these machines would be through RemoteIT on port TCP 799. This is the product the XYZ Widget Corp uses to control all remote systems. The 3rd Party developers do not have access to the database systems, and must coordinate table loads and dumps through XYZ.

Once the systems were back online, and the 3rd party developers had pushed the test web site to the Linux system, the test environment was tested, both by the developers, and by XYZ employees. Once everything was OK, XYZ's CIO told the developers that they could continue working on the test systems in a normal fashion.

As part of our final re-architecture and analysis report that I drafted for XYZ Widget Corp, we strongly recommended that they create a policy for rolling, and reviewing log files. Linux, NT, and WinRoute were now generating detailed log files, and we offered to help XYZ Widget Corp. assess 3rd party software packages to review these logs regularly. Regrettably, XYZ did not take us up on that offer (they felt that they didn't have the administrative man power to support such a project), but they have contracted with us to perform regular vulnerability assessments.

In addition to the vulnerability assessments, XYZ Widget Corp. also contracted us to perform hardening on the rest of their systems. All four Linux production systems, and the two NT production systems have all been hardened according to the SANS step-by-step guides.

Follow Up / Lessons Learned

Now that XYZ Widget Corp's developers were back to work, and all systems within their data center were properly hardened, I sat down to produce a report outlining both the details about how we handled the incident, and also a list of findings and recommendations stemming from the incident.

For starters, we felt very strongly that since there was no way of knowing for certain if the code on the production systems was pure (aka. not contaminated through the injection of trojanized code within the test systems), that XYZ Widget Corp not consider themselves safe. We felt that XYZ's administrators were now in control of the operating

systems, but we couldn't be sure of the integrity of the code itself. Our fear came from the fact the we weren't sure if we could trust that the 3rd party developers had not moved code from the testing environment to the development environments during the time that the intruder had control of the site.

Many of the other recommendations that came from this engagement were handled during various phases of the incident, such as the switching from FTP to SSH as a vehicle for the developers to use to deliver code updates to the test system for testing, the hardening of all systems, and the increased logging capabilities of all systems.

Along with the changes already made to XYZ Widget Corp's environment, we made several additional recommendations that we felt would provide a better overall security posture.

1. The largest, and most obvious change involves the removal of the three bastion systems from the exterior segment to a more secure location behind the firewall. This is a good way to consolidate log files since filtering would now be done on the firewall instead of on the individual systems. Of course there's nothing wrong with keeping the system level filtering, if it isn't hurting performance.
2. Once the bastion systems are moved to the DMZ, add port filtering to the Alteon switch to assure that no access is allowed between the test systems and production systems.
3. Incorporate encryption into the mail client. Another concern we had was whether or not the intruder was reading corporate email while he had control. It was impossible for XYZ Widget Corp. to fully assess how much proprietary information could have been lost in this fashion. We recommended that they purchase corporate versions of PGP software.
4. XYZ Widget Corp. needs to perform, or higher a consultant to perform, a thorough code review to assure the integrity of their production source code.

We believe that these few changes will go far in improving XYZ's security posture. As of the writing of this report, XYZ Widget Corp. had begun implementing these environment changes, and policy and procedure for the moving of code between systems were in the process off being ironed out to assure a good, clean audit trail for future code migrations.

There were a number of important lessons that can be learned from this incident. The most obvious is how important it is to properly harden Internet hosts. The need to properly defend mission critical systems is tantamount to tweaking performance and providing functionality. If a system is attacked and data is lost, not only does a company potentially face legal retribution from their client base, but there integrity and consumer confidence will most surly be damaged with even the smallest bit of negative publicity. XYZ Widget Corp. learned that security is just as much about saving face as it is about saving data.

We also learned the importance of strict policy and procedure and adherence to these policies and procedures. We were faced with the unfortunate scenario where we're not sure if we can truly trust the integrity of the source code. Contamination may have occurred during interchanges of data between the developers and the test servers. This uncertainty has forced us to consider this site at risk, until such time as a proper code review can be done, and these fears can be dispelled.