

# AntiOnline.com

Hackers Know The Weaknesses In your System. Shouldn't You?

# Newsletter #7

# Table of Contents

Editorial  
by MsMittens .....p. 3

Zero-Day Exploits  
by Tony Bradley.....p. 4

Book Review: O'Reilly Linux Network Administrator's  
Guide  
by hatebreed2000.....p. 6

Pluggable Authentication Modules (PAM)  
by MsMittens.....p. 10

Pandora  
by Tony Bradley.....p. 15

Winter Class of 2003: War Games Report  
by MsMittens.....p. 17

# MsMittens' Editorial

Well, here's our first issue produced under the new management. This year's start has been relatively quiet. The biggest change has been the introduction of the SuperDMCA's and, of course, the thing on everyone's mind has been the War.

The SuperDMCAs have yet to be fully tested. But there has been much broo-ha-ha about the laws proposed and the risk they hold not for attackers but rather for administrators. I wonder if they will really be ever tested. Operating systems are being more locked down and secured by manufacturers, albeit they aren't perfectly locked down. Larger attacks that were highlighted in the mid-late 1990s seem to be a thing of the past. Maybe attackers are hiding more.

I remember reading somewhere that security administrators were fighting a losing battle. But is that true? Yes, there are still attacks that happen but many of the attacks are smaller scale than what was seen before and certainly are not seen by the public. Perhaps the battle we are losing is complacency rather than a battle of "attacker" versus "admin".

Ours is an odd community. We are a mish-mash of admin, security geeks, educators, students, wannabes and everything in between. I think we can ask enough important – and even controversial at times – questions that will keep us on our toes to deal with attacks before, during and after they happen. We are still willing to learn from our mistakes. The question is: what about everyone else? Maybe they truly do not have anything worthwhile to steal – except who they are.

On that note, I wanted to comment about our new management. It has been an interesting time with the new management changeover. And as they get to understand our unique community, I think they will be able to make changes that will benefit the community. I always believe in giving people a chance and I think that Jupitermedia does deserve that chance. And this is not just because they want to keep the newsletter as it is and potentially expand it to more (not talking ads here, articles). And it's not just because they've also assigned Negative and myself to larger moderator roles on the board.

But it is because they don't want to change much of AntiOnline. They like the community as it is.

Times change. People change. But certain ideals are set in stone. To me, one ideal of AntiOnline that I hold true and keep written in stone is this and is a bit of an ideal from *The Art of War*: for me to be a better administrator I need to have the necessary tools to fight the war against the attacker. And if I understand how my enemy fights, then I understand how to defend and defeat him.

Enjoy the newsletter. :)

# Zero-Day Exploits

## by Tony Bradley

One of the mantras of information security is to keep your systems patched and updated. As vendors learn about new vulnerabilities in their products, either from 3<sup>rd</sup>-party researchers or through their own discoveries, they create hotfixes, patches, service packs and security updates to repair the holes.

The Holy Grail for malicious program and virus writers is the "zero-day exploit". A zero-day exploit is when the exploit for the vulnerability is created before, or on the same day as the vulnerability is learned about by the vendor. By creating a virus or worm that takes advantage of a vulnerability the vendor is not yet aware of and for which there is not currently a patch available the attacker can wreak maximum havoc.

Some vulnerabilities are dubbed zero-day exploit vulnerabilities by the media, but the question is zero-day by whose calendar? Often times the vendor key technology providers are aware of a vulnerability weeks or even months before an exploit is created or before the vulnerability is disclosed publicly.

A glaring example of this was the SNMP (Simple Network Management Protocol) vulnerability announced in February of 2002. Students at Oulu University in Finland actually discovered the flaws in the summer of 2001 while working on the PROTOS project, a test suite designed to test SNMPv1 (version 1).

SNMP is a simple protocol for devices to talk to each other. It is used for device to device communication and for remote monitoring and configuration of network devices by administrators. SNMP is present in network hardware (routers, switches, hubs, etc.), printers, copiers, fax machines, high-end computerized medical equipment and in almost every operating system.

After discovering that they could crash or disable devices using their PROTOS test suite, the students at Oulu University discreetly notified the powers that be and the word went out to the vendors. Everyone sat on that information and kept it secret until it was somehow leaked to the world that the PROTOS test suite itself, which was freely and publicly available, could be used as the exploit code to bring down SNMP devices. Only then did the vendors and the world scramble to create and release patches to address the situation.

The world panicked and it was treated as a zero-day exploit when in fact more than 6 months went by from the time the vulnerability was originally discovered. Similarly, Microsoft finds new holes or is alerted to new holes in their products on a regular basis. Some of them are a matter of interpretation and Microsoft may or may not agree that it is actually a flaw or vulnerability. But, even for many of the ones they agree are vulnerabilities there could be weeks or months that go by before Microsoft releases a security update or service pack that addresses the issue.

One security organization ([PivX Solutions](#)) maintains a running list of Microsoft Internet Explorer vulnerabilities that Microsoft has been made aware of but haven't yet patched ([Unpatched IE Security Holes](#)). There are other sites on the web frequented by hackers that maintain lists of known vulnerabilities and where hackers and malicious code developers trade information as well.

This is not to say that the zero-day exploit doesn't exist. Unfortunately it also happens all too often that the first time the vendors or the world are made aware of a hole is when doing a forensic investigation to find out how a system was broken into or when analyzing a virus that is already spreading in the wild to find out how it works.

Whether the vendors knew about the vulnerability a year ago or found out about it this morning, if the exploit code exists when the vulnerability is made public it's a zero-day exploit on your calendar.

The best thing you can do to protect against zero-day exploits is to follow good security policies in the first place. By installing and keeping your anti-virus software up to date, blocking file attachments to emails which may be harmful and keeping your system patched against the vulnerabilities you are already aware of you can secure your system or network against 99% of what is out there.

One of the best measures for protecting against currently unknown threats is to employ a hardware or software (or both) firewall. You can also enable heuristic scanning (a technology used to attempt to block viruses or worms that are not yet known about) in your anti-virus software. By blocking unnecessary traffic in the first place with a hardware firewall, blocking access to system resources and services with a software firewall or using your anti-virus software to help detect anomalous behavior you can better protect yourself against the dreaded zero-day exploit.

# O'Reilly Linux Network Administrators Guide

## written by hatebreed2000

This book has been around for awhile but I had never read it up until this point. First let me state that this book is great for newbies in many ways, unfortunately at some points of this book it gets a bit confusing if you do not have a basic background knowledge of some applications such as one of the chapters regarding the configuration of UUCP, which in my experience is not for the faint of heart. I would recommend having other resources at your disposal namely a good search engine like google in case there is something that you may not be totally confident in understanding. So with that in mind I will do a basic review of the chapters included in the book and the high and low points of each.

Chapter one starts off by discussing the history of linux and covers basic networking information on UUCP, TCP/IP, various protocols, hardware, and security. Any one who has read a O'Reilly book before knows that they always start off with one of these chapters so there's really not much to say about this.

Chapter two address issues of TCP/IP networking, most of you I would hope already have at least a basic understanding of this protocol. If you do not then this chapter will be great for you, and you may want to read it just because you need a quick review or maybe you will find a interesting tid bit of information that you were unaware of. Either way I would suggest reading this chapter unless you feel fully confident in your knowledge of TCP/IP.

Chapter three deals with with very basic configuration of networking hardware. Such as building a kernel and setting up your Ethernet card. Up until this point the book discusses the network interfaces and general TCP/IP issues, but now it gets interesting by explaining the ways that the kernel access's the device drivers and how the different interfaces from Ethernet, FDDI to Token Ring handle the request. What I liked about this chapter was that it did not stick to just one type of protocol or interface, it explained everything from IRQ's to how I/O address are mapped. I found the chapter to be a bit redundant at times regarding configuration of the PLIP and SLIP protocols.

Chapter four discusses serial hardware configuration. I did not know much about UUCP before reading this book so this provided a great starting point. It provides good information on using the stty and setserial commands and talks about the mgetty daemon. The only complaint I have about this chapter is that it is almost too much of a newbies guide. If your anything like me then just a little information on a subject is not enough, you have got to have more. So this is one of those time that the internet to search on whatis.com or whatever would come in very handy.

Chapter five disusses the TCP/IP network configuration. This chapter I thought was great, it started off by providing some good information on how to mount the /proc file system, and

moved on to installing binaries and setting the hostname. This chapter provided a great background on sub-netting and configuration of gateways. The `ifconfig` command as well as the `netstat` commands were discussed in depth and provided some very good information, as well as briefly discussing the ARP tables.

Chapter six talks about the name service and resolver configuration. This chapter was mostly about `.conf` files such as the `host.conf` and `resolv.conf` files among many others, the writer did an excellent job in providing how to config the files to suit your needs as well as providing illustrations to help you along. The chapter also included a good theory of operation illustration on how DNS works and discussed in depth the different flags and syntax of the `nslookup` command, which for those of you who have some ambition on becoming "elite" is a very good read due to the fact that this is where a lot of footprinting begins.

Chapter seven explains the serial line IP. For those of you on a tight budget, dial-up is the way to go. The low cost and availability of dial-up is appealing to some but obviously it is not for a large network or for those who do online gaming, but if all you do is surf the net then this may be an option for you. All you need is a modem and a serial port equipped with a FIFO buffer to run your dial-up connection. The SLIP and PPP setup can be troublesome for some but I'm sure that after reading this book you will have no problem setting up a connection.

Chapter eight discusses the Point-TO-Point protocol. This chapter was not that good due to the fact that a lot of the information provided is outdated. However it does provide a good overview of the protocol and the advantages and disadvantages of PPP over SLIP as well as giving a decent compare and contrast of CHAP and PAP.

Chapter nine is talks about the TCP/IP firewall. The chapter starts off by taking you into the mind of the attacker and prides some common methods of attacks including DoS and spoofing. This chapter is great for those of you who don't fully understand the different types of firewalls and providing good illustrations on how they protect and serve the network. The most interesting and important part of this chapter though is the discussions about the `IPfwadm`, `IPchains`, and the newest and best yet `IPtables` utilities. I also found the part on setting TOS bits very noteworthy just because I had no knowledge in the area.

Chapter ten is called IP accounting. In a network setting it is very important to know just how much data you are transmitting and receiving over your network, so for those of you with even a small LAN you will not want to skip this chapter. The writer does an exceptional job on illustrating how to config the kernel for accounting IP's and giving an in depth talk on the different ways to config the kernel by way of service port, by address and by protocol among others.

Chapter eleven talks about masquerading and NAT. This chapter I thought could have been a lot more in depth, but still, it provides a good starting place for you to configure the kernel to use network masquerading, while giving some need to know network address translation information such as how to use `IPtables` to generate NAT rules that map just about anything, with combinations of matches using any of the standard attributes, such as source address, destination address, protocol type and port number.

Chapter twelve address important network features. This chapter is very important to any

serious network admin. It includes topics such as configuring SSH, the inetd super server, RPC's and the many issues that come with remote logins. The discussion on SSH is very indepth, it provides information on installing and configuration as well as how to secure it. The topic of RPC's is very good, although could have been a bit longer, but nevertheless provides crucial points on the do's and don'ts.

Chapter thirteen talks about the network information system. This chapter talks about how to configure your network to seem as a single computer to your users, the NIS was developed by sun systems and was based on RPC, so the goal of this chapter at times seems more of a history lesson more than a technical how-to. It basically just gets you comfortable with knowing your way around the NIS as well as talking about the pro's and con's of NIS and NIS+. It provides some basic information on setting up a NIS client with GNU libc, as well as explaining well how to secure both.

Chapter fourteen is all about the network file system. The network file system is probably the most prominent network service using RPC, it allows you to access files on a remote host in exactly the same way you would locally. So with just knowing that I'm sure you can see how important this chapter is. The writer did a very good job in providing all the necessary information needed on how to prep the NFS and how to mount it. NFS daemons is discussed very briefly, even providing some sample boot scripts for you to play with if you so choose. The exports file is one of the most important files in your NFS and as you read you will become very clear of this.

Chapter fifteen talks about the IPX and NCP filesystems. I really liked this chapter because these protocols have been around for a very long time....and they are still in use. Way before Microsoft was the giant it is, it was Xerox and Novell and the writer makes it a point to discuss this, while still not straying from the topic at hand. This is probably the longest chapter in the book, and for good reason, it talks about everything from how to configure IPX under linux to managing print queues. For those of you with no knowledge of Novell this chapter will open your eyes, it will give you a whole new respect for those that sit at a sever for hours and hours just configuring the server. Because those of you that have configed a novell server know that it is no simple task, and the task involved in doing so are only briefly discussed in this chapter due to the fact that it is a bit of a feat to set up a secure Novell server. My advise to those of you that get this book and read it is, after you get done reading it, is dont just stop with this chapter, thinking that you got enough information regarding Novell, I suggest you hope on the net and keep reading.

Chapter sixteen is on managing Taylor UUCP. Despite the popularity of dial-up PPP and SLIP connections to the internet, many people still use UUCP due to how cheap it is and because of how old it is, its the only connection offered in some countries. Topics in this chapter include configuration (DUH!! :P) the inner workings of uucico and how to use it on the command line. As you read you will quickly see that the configuration of UUCP is not easy, to put it nicely....it is a bitch. Taylor UUCP however, is somewhat easier. Hopefully after reading this chapter you will have at least somewhat of a idea on what it takes to set a UUCP connection up, more than likely though, the average user will never have to config a UUCP connection, but by the chance you would, I would not just settle for this book as a reference. Do not be afraid to use the library or google if you do not understand something.

Chapter seventeen is all about E-mail. I started this chapter thinkin that there really wouldnt be much to say about electronic mail, but I soon relized that there is quite a bit to say about it under linux. The topics range from RFC 822 to mail formates as well as routing mail on the internet and discuusion of a tool i had never heard of until I read this book, a tool called Elm. It does not explain much on how to use Elm, instead it concentrates on how to configure it.

Chapter eighteen is on Sendmail. The writer does a exceptional job on explaining the sendmail.cf and sendmail.mc files and providing some sample config files as well as providing the parameters used in each. Sendmail has alot to configure so again I wouldnt settle with just this book as a reference, however it does touch on all the impotent topics such as defining mail transport protocol, interpreting and weiting rewrite rules and macro defintions. I especially liked how they included a section dedicated to spam and how to filter it.

Chapter nineteen discusses Exim. This application is largely compatiabile with sendmail but the config files are totally different. Many of the topics that are discussed in this chapter are the same as the ones addressed in the sendmail chapter.

Chapter twenty talks about netnews. Usenet.....need I say more?

Chapter twentyone is on Cnews. It was developed for UUCP connections and this chapter describes how to install and properly configure Cnews under linux.

Chapter twentytwo talks about NNTP and the nntp daemon. This chapter is pretty self explainitory, topics include how to connect to the news server and how to deliver news, as well as different delivery options.

Chapter twentythree is about internet news. The chapter explains howto install and configure INN, as well as giving a exaustive list of options to use under the configuration file. I was suposed at some of the simple yet effective tips that this chapter provided on maintaing your INN.

Chapter twentyfour discusses how to configure newsreader. Newsreader is a program that users can use to view, store, and write articles. There are many of these that have been ported to linux but only the three most popular are discussed. Those three are tin, trn, and nn. This chapter had a particular nostalgic feel to it, one that I myself could not relate to since I have only been into computers for the past 3 years. Nevertheless I found it informative but not very useful.

I hope this review was of some help to you the reader, looking for a good book to pick up. As I said before, it is not for advanced readers but for those of you who know your stuff but your not quite up to that uber-user stage yet. So if your one of those that thinks this book sounds good and would like to read it, which I recomend, let me know as I have the book in a .pdf format.

hatebreed2000

# Pluggable Authentication Modules (PAM)

by MsMittens

Unix local authentication and security was enhanced with the introduction of PAM (no.. not the cooking spray). Previously, authentication was merely done by user and password combination, if at all. Basically, the password was checked against the encrypted one in `/etc/passwd`. As time has shown, this method was not the best nor the most secure.

The \*nix community decided that it might be worthwhile to deal with this. Other authentication methods were designed and suggested but many required a complete rewrite of existing tools, an incorporation completely new passwd mechanism or required specific hardware implementations. Many of these presented limitations in regards to universal authentication and control over application usage.

PAM was created by Sun Microsystems in December 1995 and released as part of the Open Software Foundation. PAM allows flexibility for administrators to set authentication policies for various PAM-aware applications without recompilation due to its modular nature. PAM determines which modules are called by looking in `/etc/pam.d/` directory.

To understand a little about how PAM works we need to understand the various components of PAM. These components are used to determine how modules will be used and to what requirement level.

First, let's look at the four descriptors. Although they are sometimes referred as modules, I prefer to refer to them as descriptors so as to avoid confusion with the actual binary modules. These descriptors identify how the module will be used and for what stage of authentication it will address. The four descriptors are:

## Four PAM descriptors

**auth** – related to user authentication e.g., when asking the user to enter a password for login

**account** – related to user account management; verifying privileges

**password** – related to password management; e.g., updating of password in /etc/shadow

**session** – deals with session management e.g., logging information about the user login session to system log

As you'll see in the example at the end of this article, the usage of auth often found at the beginning as this deals with the initial login and authentication process as well as events or procedures that would happen as the user logs in (note: the checking of email).

Now, for each descriptor we have to determine how "important" that stage is. Each descriptor is assigned a flag. Based on the flag, this will tell the application if the user has provided enough correct information to proceed to the next descriptor.

One of the important thing about the flags is that if there is a failure the user is notified of the failure but it's not indicated exactly where the failure is. This means providing minimal information out, especially in regards to attackers. The four flags of PAM are:

### Four Flags for PAM Modules

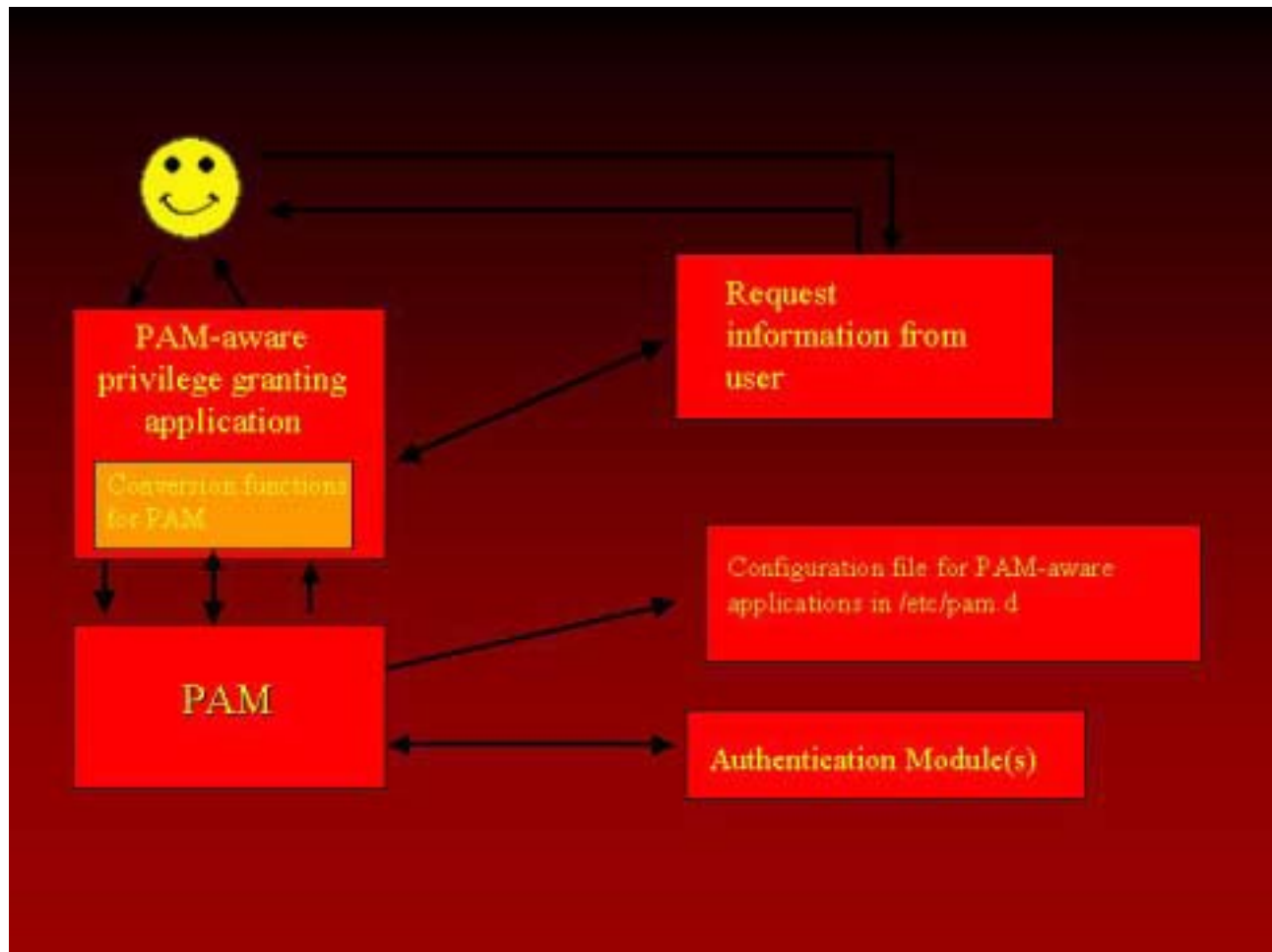
**requisite** – the action **MUST** be completed successfully to continue processing actions. If not, the service request will fail and no more action will be processed

**required** – the action must be completed successfully. If not, the services will fail **AFTER** the reset of the action has been processed

**sufficient** – action alone is enough to be accepted unless an earlier **REQUIRED** action failed. If the request is accepted based on successful action, no further actions will be processed

**optional** – if not other action has proven successful or unsuccessful, then the success of this action – or lack thereof – will determine the response to authentication request

I hope this gives you a better understanding of some of at least one method of authentication control on \*nix systems. If you intend on playing with your pam config files remember to make backups **before** you alter them. JStep-by-Step how PAM (general concept) works:



1. User invokes an application to access services
2. PAM-aware application calls PAM library to perform the authentication
3. PAM library looks up application specific configuration file in /etc/pam.d. The file tells PAM what type of authentication is required for the application
4. PAM library loads required authentication modules

5. Modules make PAM communication with the conversation functions available in the application
6. Conversion function requests information from the user e.g., password, retina scan
7. User provides requested information
8. PAM authentication modules provide authentication status via library
9. If successful then:
  - grants the requested privilege
  - informs the user the process has failed

**Specific Sample: /etc/pam.d/login**

<b>auth</b>	<b>required</b>	<b>pam_securetty.so</b>
<b>auth</b>	<b>required</b>	<b>pam_pwdb.so</b>
<b>auth</b>	<b>required</b>	<b>pam_nologin.so</b>
<b>auth</b>	<b>optional</b>	<b>pam_mail.so</b>
<b>account</b>	<b>required</b>	<b>pam_pwdb.so</b>
<b>session</b>	<b>required</b>	<b>pam_pwdb.so</b>
<b>session</b>	<b>required</b>	<b>pam_lastlog.so</b>
<b>password</b>	<b>required</b>	<b>pam_pwdb.so</b>

1. The pam\_securetty.so module checks to see that the requested user is allowed to log in at the console in question by comparing the user's login location against the /etc/securetty file. This action is required and if it fails, the authentication request will be rejected after all other actions have been completed
2. The pam\_pwdb.so module is called to see if the user has entered the corrected password (flag=required)
3. The pam\_nologin.so module is called to see whether the file /etc/nologin exists. If so the file is displayed and the action fails, preventing the user from logging in. (flag=required)
4. The pam\_mail.so is called to see if the user has any new mail (flag=optional) so the user will be allowed to log in based on the results of other actions, whether or not any new mail is present

- 5.The pam\_pwd.so is called again, this time in account context, which causes it to check for password or account expiration. If the account has expired or if the user's pass has expired and the user refuses to enter a new password the action will fail (flag=required)
- 6.The pam\_pwd.so is called again, this time in session context, causing it to enter login attempt to syslog (flag=required) NB: if system can not record log then user cannot log in
- 7.The pam\_lastlog.so module is called to update the user login history in /var/log/lastlog file
- 8.The pam\_pwd.so is called one last time to replace the user's pass in /etc/shadow, in case the password is being updated (as per step 5) during the process (flag=required)
- 9.User logs in or does not

Resources/References:

Linux System Security, 2<sup>nd</sup> Edition

Red Hat Linux Administration

<http://www.kernel-panic.org/index.pl/authentication#SECTION03000000000000000000>

<http://www.tldp.org/HOWTO/User-Authentication-HOWTO/x101.html>

**Become famous! Send an article into the AONewsletter!!**

**We want articles on security tools, "hacking" tools, ideas, rants, raves, reviews, etc.**

**Send a note to [msmittens@msmittens.com](mailto:msmittens@msmittens.com) with AONewsletter in the subject line or send a private message to my AO account, MsMittens.**

**Next deadline: June 27, 2003**

# **Pandora's Box**

## **by Tony Bradley**

On August 6, 1945 the United States released "Little Boy", an atomic bomb, on the Japanese city of Hiroshima. The bomb devastated everything for miles and left a lingering radiation that caused problems long after the initial blast. This attack led to the surrender of Japan and expedited the end of World War II, but at what cost?

As we sit at the end of the military action between the United States and Iraq, another significant weapon lays waiting for its first action. This weapon can help bring the infrastructure of enemy nations to its feet by eliminating electricity and telephone communications, knocking out radar systems and disrupting critical infrastructure.

This weapon is Cyber Warfare. The question is, what sort of Pandora's box will we open if we unleash it? National Security Presidential Directive 16 was signed by President Bush in July of 2002. The directive calls for a national policy on the rules of engagement for using Cyber Warfare as a weapon.

It has long been discussed as a potential weapon, but until now there were no established guidelines. Bush's Cyber Warfare directive gives the United States clear direction for what circumstances warrant the use of Cyber Warfare, who should authorize and execute the attack and what types of targets are allowable.

One of the biggest problems with condoning Cyber Warfare is the size of the bull's-eye on our back. The United States arguably relies on computer technology for its core infrastructure more than any other nation in the world. We also rely heavily on the Internet and the World Wide Web for every day functionality, business and commerce. Should any country successfully launch a Cyber Warfare attack against the United States it could quickly cripple the nation's ability to function.

Being the country that created the rules of engagement, and possibly the country to first use Cyber Warfare as an official military tool would mark us as a target much the same way being the fastest gun did in the Old West. Hackers around the world will consider it an extraordinary triumph to be the ones to use this weapon against the United States.

Another issue with unleashing worms and viruses intent on destroying the enemy is that the possibility exists that with a few incorrect lines of programming code the attack could spread world-wide instead of being contained to its intended target.

A biological virus can mutate over time to be resistant to the antibodies created to destroy it. It can get stronger and constantly find new ways to exploit flaws in the human immune system.

Similarly, enemy nations and hackers around the world could intercept the code from our cyber-attack once unleashed. They can analyze it, examine it, learn from it and then rebuild it different, better and stronger than before and send it back out. This happens already with everyday computer viruses.

While the ability to disable a nation's infrastructure and cripple its military defenses without firing a shot or launching a single cruise missile sounds appealing, condoning and launching Cyber Warfare is a slippery slope we may not want to start down. The United States should think very carefully about the precedent they set and the repercussions of their actions before choosing to unleash this new weapon.

**Become famous! Send an article into the AONewsletter!!**

**We want articles on security tools, "hacking" tools, ideas, rants, raves, reviews, etc.**

**Send a note to [msmittens@msmittens.com](mailto:msmittens@msmittens.com) or send a private message to my AO account, MsMittens, with AONewsletter in the subject line.**

**Next deadline: June 27, 2003**

# Winter Class of 2003: War Games Report by MsMittens

I wanted to do a small report on the war games from this past semester. It was a thoroughly enjoyable and fun experience for both myself and my students.

Every semester that I teach the **Introduction to Internet Security**, I allow the students a month of war games. The war games are tied into their final assignment. Basically they have the choice of one of "goals";

1. White Hat Attacker/Security Auditor: The goal is to attempt to break into a classmate's machine and gain access. Enough to get a "flag" (file designated by the "victim" and hidden by the victim – the file is usually named *victimflag*). Students can use any tool.\*
2. Same as above but instead of a classmate they attempt to break into "Tank", our "security" server. This box, built on a RH6 installation, is so locked down we've had few break-ins. I managed one this year with a newer ptrace but that was soon solved.
3. Security Administrator. How well can the student lock down their box against attack and what steps did they take to do it? This one is often done by students who favour the Windows environment and they take it incredibly seriously. I've had reports handed in that detailed every registry key changed, accounts move, permissions altered, etc. They learn quickly that a firewall just doesn't cut it.
4. Lastly, all students had to have 3 services running: FTP, Web and another of their choice. Part of this was to mimic real life, internet access (as well as to

give "attackers" something to go after!) It also limited the usage of firewalls that dropped everything. ;)

\*one exception that I had to put in place was the limitation of DDoSes and DoSes. The reason was that our router, which happened to be an older Unix box, was quite unstable. It wasn't uncommon for us to use even ettercap and watch the network crash.

For the most part, students went whole-hearted into it. I had to kick out students from other classes for fear they'd lose their installations. All students in the class started with a dual boot: usually a Linux of their choice (not too new) as well as a Windows of their choice (usually 2000 but there was a few NT, an ME and an XP installations). They configured their installation of choice for the war games a week or two before (if so inclined). One of the rules of the war games is that you can only use one of the two.

The students that were successful at the war games were the ones that buddied up or formed teams. Some of the teams included "spies", from what I could see. The "spies" would go to the victim and talk to them about what they were doing, giving advice. If the victim ever left to go to the washroom or for a smoke but didn't log out, the "spy" would quickly put an account on the machine (usually some variation of root named as mial) and then act like nothing had happened. The rest of the team would then.. well.. take over the machine. One group had managed to "infiltrate" about 5-6 other student machines. Quite impressive (keep in mind that these are networking students with limited programming experience.)

The other reason for their success was the fact that they paid attention to something I had taught them early on: attackers plan out how they will attack. The successful students followed a methodology that was simple: choose a target, gather information about the target and their system(s), find vulnerabilities they may be susceptible to, attack, elevate privileges, go stealth by altering logs and leaving backdoors. Since many of the students do not have a programming background (this is a networking course, with strong emphasis on networking processes themselves, and programming is limited to perl and simpler web languages), they became quite proficient at finding exploits in the wild.

As for me, I got "caught" by one student on Tank. I tend to be curious as to what my students work on and played with a tool a student left in */tmp*. Bad mistake on my part. The student had modified a little hack he had found so that when the victim ran the script, it replaced the *.bash\_profile*. Very good for someone who had no C experience. He was kind enough to send me the file so I could see it. With a little more experience, he could have modified it so that it might have gotten more than just replacing the profile file.

And for those curious, the modification of my *.bash\_profile* was this:

```
echo "OsAmA OwNs YOU"; sleep 15; exit
```

Thankfully, sftp doesn't rely on *.bash\_profile*. J

Anyways, that's just the general report. What had my students learned from all this? Well, they learned how powerful social engineering is to the attacker. They learned how to be paranoid. And they learned how important it is to be vigilant. For a lot of them, it was a chance to see what kind of career they might consider beyond a simple administrator.

**Become famous! Send an article into the AONewsletter!!**

**We want articles on security tools, "hacking" tools, ideas, rants, raves, reviews, etc.**

**Send a note to [msmittens@msmittens.com](mailto:msmittens@msmittens.com) or send a private message to my AO account, MsMittens, with AONewsletter in the subject line.**

**Next deadline: June 27, 2003**