T1040 - Network Sniffing

Description from ATT&CK

> Adversaries may passively sniff network traffic to capture information about an environment, including authentication material passed over the network. Network sniffing refers to using the network interface on a system to monitor or capture information sent over a wired or wireless connection. An adversary may place a network interface into promiscuous mode to passively access data in transit over the network, or use span ports to capture a larger amount of data. Data captured via this technique may include user credentials, especially those sent over an insecure, unencrypted protocol. Techniques for name service resolution poisoning, such as LLMNR/NBT-NS Poisoning and SMB Relay, can also be used to capture credentials to websites, proxies, and internal systems by redirecting traffic to an adversary.
>
> Network sniffing may reveal configuration details, such as running services, version numbers, and other network characteristics (e.g. IP addresses, hostnames, VLAN IDs) necessary for subsequent Lateral Movement and/or Defense Evasion activities. Adversaries may likely also utilize network sniffing during Adversary-in-the-Middle (AiTM) to passively gain additional knowledge about the environment.

> In cloud-based environments, adversaries may still be able to use traffic mirroring services to sniff network traffic from virtual machines. For example, AWS Traffic Mirroring, GCP Packet Mirroring, and Azure vTap allow users to define specified instances to collect traffic from and specified targets to send collected traffic to.(Citation: AWS Traffic Mirroring)(Citation: GCP Packet Mirroring) (Citation: Azure Virtual Network TAP) Often, much of this traffic will be in cleartext due to the use of TLS termination at the load balancer level to reduce the strain of encrypting and decrypting traffic.(Citation: Rhino Security Labs AWS VPC Traffic Mirroring)(Citation: SpecterOps AWS Traffic Mirroring) The adversary can then use exfiltration techniques such as Transfer Data to Cloud Account in order to access the sniffed traffic.(Citation: Rhino Security Labs AWS VPC Traffic Mirroring)
>
> On network devices, adversaries may perform network captures using Network Device CLI commands such as `monitor capture`.(Citation: US-CERT-TA18-106A)(Citation: capture_embedded_packet_on_software)

Atomic Tests

- Atomic Test #1 - Packet Capture Linux using tshark or tcpdump
- Atomic Test #2 - Packet Capture FreeBSD using tshark or tcpdump
- Atomic Test #3 - Packet Capture macOS using tcpdump or tshark
- Atomic Test #4 - Packet Capture Windows Command Prompt
- Atomic Test #5 - Windows Internal Packet Capture
- Atomic Test #6 - Windows Internal pktmon capture
- Atomic Test #7 - Windows Internal pktmon set filter
- Atomic Test #8 - Packet Capture macOS using /dev/bpfN with sudo
- Atomic Test #9 - Filtered Packet Capture macOS using /dev/bpfN with sudo
- Atomic Test #10 - Packet Capture FreeBSD using /dev/bpfN with sudo
- Atomic Test #11 - Filtered Packet Capture FreeBSD using /dev/bpfN with sudo
- Atomic Test #12 - Packet Capture Linux socket AF_PACKET,SOCK_RAW with sudo
- Atomic Test #13 - Packet Capture Linux socket AF_INET,SOCK_RAW,TCP with sudo

Atomic Test #1 - Packet Capture Linux using tshark or tcpdump

Perform a PCAP. Wireshark will be required for tshark. TCPdump may already be installed.

Upon successful execution, tshark or tcpdump will execute and capture 5 packets on interface ens33.

Supported Platforms: Linux

auto_generated_guid: 7fe741f7-b265-4951-a7c7-320889083b3e

Inputs:

Name	Description	Type	Default Value
interface	Specify interface to perform PCAP on.	string	ens33

Attack Commands: Run with `bash`! Elevation Required (e.g. root or admin)

```
tcpdump -c 5 -nnni #{interface}
tshark -c 5 -i #{interface}
```

Dependencies: Run with `bash`!

Description: Check if at least one of tcpdump or tshark is installed.

Check Prereq Commands:

```
if [ ! -x "$(command -v tcpdump)" ] && [ ! -x "$(command -v tshark)" ]; then exit :
```

Get Prereq Commands:

```
(which yum && yum -y install epel-release tcpdump tshark)||(which apt-get && DEBIAI
```

Atomic Test #2 - Packet Capture FreeBSD using tshark or tcpdump

Perform a PCAP. Wireshark will be required for tshark. TCPdump may already be installed.

Upon successful execution, tshark or tcpdump will execute and capture 5 packets on interface ens33.

Supported Platforms: Linux

auto_generated_guid: c93f2492-9ebe-44b5-8b45-36574cccfe67

Inputs:

Name	Description	Type	Default Value
interface	Specify interface to perform PCAP on.	string	em0

Attack Commands: Run with `sh`! Elevation Required (e.g. root or admin)

```
tcpdump -c 5 -nnni #{interface}
tshark -c 5 -i #{interface}
```

Dependencies: Run with `sh`!

Description: Check if at least one of tcpdump or tshark is installed.

Check Prereq Commands:

```
if [ ! -x "$(command -v tcpdump)" ] && [ ! -x "$(command -v tshark)" ]; then exit :
```

Get Prereq Commands:

```
(which pkg && pkg install -y wireshark-nox11)
```

Atomic Test #3 - Packet Capture macOS using tcpdump or tshark

Perform a PCAP on macOS. This will require Wireshark/tshark to be installed. TCPdump may already be installed.

Upon successful execution, tshark or tcpdump will execute and capture 5 packets on interface en0A.

Supported Platforms: macOS

auto_generated_guid: 9d04efee-eff5-4240-b8d2-07792b873608

Inputs:

Name	Description	Type	Default Value
interface	Specify interface to perform PCAP on.	string	en0A

Attack Commands: Run with `bash`! Elevation Required (e.g. root or admin)

```
sudo tcpdump -c 5 -nnni #{interface}
if [ -x "$(command -v tshark)" ]; then sudo tshark -c 5 -i #{interface}; fi;
```

Dependencies: Run with `bash`!

Description: Check if at least one of tcpdump or tshark is installed.

Check Prereq Commands:

```
if [ ! -x "$(command -v tcpdump)" ] && [ ! -x "$(command -v tshark)" ]; then exit :
```

Get Prereq Commands:

```
(which yum && yum -y install epel-release tcpdump tshark)||(which apt-get && DEBIAI
```

Atomic Test #4 - Packet Capture Windows Command Prompt

Perform a packet capture using the windows command prompt. This will require a host that has Wireshark/Tshark installed.

Upon successful execution, tshark will execute and capture 5 packets on interface "Ethernet".

Supported Platforms: Windows

auto_generated_guid: a5b2f6a0-24b4-493e-9590-c699f75723ca

Inputs:

Name	Description	Type	Default Value
interface	Specify interface to perform PCAP on.	string	Ethernet
wireshark_url	wireshark installer download URL	url	https://1.eu.dl.wireshark.org/win64/Wireshark-latest-x64.exe
tshark_path	path to tshark.exe	path	c:\program files\wireshark\tshark.exe
npcap_url	npcap installed download URL	url	https://nmap.org/npcap/dist/npcap-1.31.exe
npcap_path	path to npcap.sys	path	C:\Program Files\Npcap\npcap.sys

Attack Commands: Run with `command_prompt`! Elevation Required (e.g. root or admin)

```
"c:\Program Files\Wireshark\tshark.exe" -i #{interface} -c 5
```

Dependencies: Run with `powershell`!

Description: tshark must be installed and in the default path of "c:\Program Files\Wireshark\Tshark.exe".

Check Prereq Commands:

```
if (test-path "#{tshark_path}") {exit 0} else {exit 1}
```

Get Prereq Commands:

```
New-Item -Type Directory "PathToAtomicsFolder\..\ExternalPayloads\" -ErrorAction I
Invoke-WebRequest -OutFile "PathToAtomicsFolder\..\ExternalPayloads\wireshark_inst
Start-Process "PathToAtomicsFolder\..\ExternalPayloads\wireshark_installer.exe" /S
```

Description: npcap must be installed.

Check Prereq Commands:

```
if (test-path "#{npcap_path}") {exit 0} else {exit 1}
```

Get Prereq Commands:

```
New-Item -Type Directory "PathToAtomicsFolder\..\ExternalPayloads\" -ErrorAction I
Invoke-WebRequest -OutFile "PathToAtomicsFolder\..\ExternalPayloads\npcap_installe
Start-Process "PathToAtomicsFolder\..\ExternalPayloads\npcap_installer.exe"
```

Atomic Test #5 - Windows Internal Packet Capture

Uses the built-in Windows packet capture After execution you should find a file named trace.etl and trace.cab in the temp directory

Supported Platforms: Windows

auto_generated_guid: b5656f67-d67f-4de8-8e62-b5581630f528

Attack Commands: Run with `command_prompt`! Elevation Required (e.g. root or admin)

```
netsh trace start capture=yes tracefile=%temp%\trace.etl maxsize=10
```

Cleanup Commands:

```
netsh trace stop >nul 2>&1
TIMEOUT /T 5 >nul 2>&1
del %temp%\trace.etl >nul 2>&1
del %temp%\trace.cab >nul 2>&1
```

Atomic Test #6 - Windows Internal pktmon capture

Will start a packet capture and store log file as t1040.etl. https://lolbas-project.github.io/lolbas/Binaries/Pktmon/

Supported Platforms: Windows

auto_generated_guid: c67ba807-f48b-446e-b955-e4928cd1bf91

Attack Commands: Run with `command_prompt`! Elevation Required (e.g. root or admin)

```
pktmon.exe start --etw  -f %TEMP%\t1040.etl
TIMEOUT /T 5 >nul 2>&1
pktmon.exe stop
```

Cleanup Commands:

```
del %TEMP%\t1040.etl
```

Atomic Test #7 - Windows Internal pktmon set filter

Select Desired ports for packet capture https://lolbas-project.github.io/lolbas/Binaries/Pktmon/

Supported Platforms: Windows

auto_generated_guid: 855fb8b4-b8ab-4785-ae77-09f5df7bff55

Attack Commands: Run with `command_prompt`! Elevation Required (e.g. root or admin)

```
pktmon.exe filter add -p 445
```

Cleanup Commands:

```
pktmon filter remove
```

Atomic Test #8 - Packet Capture macOS using /dev/bpfN with sudo

Opens a /dev/bpf file (O_RDONLY) and captures packets for a few seconds.

Supported Platforms: macOS

auto_generated_guid: e6fe5095-545d-4c8b-a0ae-e863914be3aa

Inputs:

Name	Description	Type	Default Value
ifname	Specify interface to perform PCAP on.	string	en0
csource_path	Path to C program source	string	PathToAtomicsFolder/T1040/src/macos_pcapdemo.c

program_path	Path to compiled C program	string	/tmp/t1040_macos_pcapdemo

Attack Commands: Run with `bash`! Elevation Required (e.g. root or admin)

```
sudo #{program_path} -i #{ifname} -t 3
```

Cleanup Commands:

```
rm -f #{program_path}
```

Dependencies: Run with `bash`!

Description: compile C program

Check Prereq Commands:

```
exit 1
```

Get Prereq Commands:

```
cc #{csource_path} -o #{program_path}
```

Atomic Test #9 - Filtered Packet Capture macOS using /dev/bpfN with sudo

Opens a /dev/bpf file (O_RDONLY), sets BPF filter for 'udp' and captures packets for a few seconds.

Supported Platforms: macOS

auto_generated_guid: e2480aee-23f3-4f34-80ce-de221e27cd19

Inputs:

Name	Description	Type	Default Value
ifname	Specify interface to perform PCAP on.	string	en0
csource_path	Path to C program source	string	PathToAtomicsFolder/T1040/src/macos_pcapdemo.c
program_path	Path to compiled C program	string	/tmp/t1040_macos_pcapdemo

Attack Commands: Run with `bash`! Elevation Required (e.g. root or admin)

```
sudo #{program_path} -f -i #{ifname} -t 3
```

Cleanup Commands:

```
rm -f #{program_path}
```

Dependencies: Run with `bash`!

Description: compile C program

Check Prereq Commands:

```
exit 1
```

Get Prereq Commands:

```
cc #{csource_path} -o #{program_path}
```

Atomic Test #10 - Packet Capture FreeBSD using /dev/bpfN with sudo

Opens a /dev/bpf file (O_RDONLY) and captures packets for a few seconds.

Supported Platforms: Linux

auto_generated_guid: e2028771-1bfb-48f5-b5e6-e50ee0942a14

Inputs:

Name	Description	Type	Default Value
ifname	Specify interface to perform PCAP on.	string	em0
csource_path	Path to C program source	string	PathToAtomicsFolder/T1040/src/freebsd_pcapdemo.c
program_path	Path to compiled C program	string	/tmp/t1040_freebsd_pcapdemo

Attack Commands: Run with `sh`! Elevation Required (e.g. root or admin)

```
sudo #{program_path} -i #{ifname} -t 3
```

Cleanup Commands:

```
rm -f #{program_path}
```

Dependencies: Run with `sh`!

Description: compile C program

Check Prereq Commands:

```
exit 1
```

Get Prereq Commands:

```
cc #{csource_path} -o #{program_path}
```

Atomic Test #11 - Filtered Packet Capture FreeBSD using /dev/bpfN with sudo

Opens a /dev/bpf file (O_RDONLY), sets BPF filter for 'udp' and captures packets for a few seconds.

Supported Platforms: Linux

auto_generated_guid: a3a0d4c9-c068-4563-a08d-583bd05b884c

Inputs:

Name	Description	Type	Default Value
ifname	Specify interface to perform PCAP on.	string	em0
csource_path	Path to C program source	string	PathToAtomicsFolder/T1040/src/freebsd_pcapdemo.c
program_path	Path to compiled C program	string	/tmp/t1040_freebsd_pcapdemo

Attack Commands: Run with `sh`! Elevation Required (e.g. root or admin)

```
sudo #{program_path} -f -i #{ifname} -t 3
```

Cleanup Commands:

```
rm -f #{program_path}
```

Dependencies: Run with `sh`!

Description: compile C program

Check Prereq Commands:

```
exit 1
```

Get Prereq Commands:

```
cc #{csource_path} -o #{program_path}
```

Atomic Test #12 - Packet Capture Linux socket AF_PACKET,SOCK_RAW with sudo

Captures packets with domain=AF_PACKET, type=SOCK_RAW for a few seconds.

Supported Platforms: Linux

auto_generated_guid: 10c710c9-9104-4d5f-8829-5b65391e2a29

Inputs:

Name	Description	Type	Default Value

csource_path	Path to C program source	string	PathToAtomicsFolder/T1040/src/linux_pcapdemo.c
program_path	Path to compiled C program	string	/tmp/t1040_linux_pcapdemo

Attack Commands: Run with `bash`! Elevation Required (e.g. root or admin)

```
sudo #{program_path} -a -t 3
```

Cleanup Commands:

```
rm -f #{program_path}
```

Dependencies: Run with `bash`!

Description: compile C program

Check Prereq Commands:

```
if [ -f "#{program_path}" ]; then exit 0; else exit 1; fi
```

Get Prereq Commands:

```
cc #{csource_path} -o #{program_path}
```

Atomic Test #13 - Packet Capture Linux socket AF_INET,SOCK_RAW,TCP with sudo

Captures packets with domain=AF_INET,type=SOCK_RAW,protocol=TCP for a few seconds.

Supported Platforms: Linux

auto_generated_guid: 7a0895f0-84c1-4adf-8491-a21510b1d4c1

Inputs:

Name	Description	Type	Default Value
csource_path	Path to C program source	string	PathToAtomicsFolder/T1040/src/linux_pcapdemo.c
program_path	Path to compiled C program	string	/tmp/t1040_linux_pcapdemo

Attack Commands: Run with `bash`! Elevation Required (e.g. root or admin)

```
sudo #{program_path} -4 -p 6 -t 3
```

Cleanup Commands:

```
rm -f #{program_path}
```

Dependencies: Run with `bash`!

Description: compile C program

Check Prereq Commands:

```
if [ -f "#{program_path}" ]; then exit 0; else exit 1; fi
```

Get Prereq Commands:

```
cc #{csource_path} -o #{program_path}
```

Atomic Test #14 - Packet Capture Linux socket AF_INET,SOCK_PACKET,UDP with sudo

Captures packets with domain=AF_INET,type=SOCK_PACKET,protocol=UDP for a few seconds. SOCK_PACKET is "obsolete" according to the man page, but still works on Ubuntu 20.04

Supported Platforms: Linux

auto_generated_guid: 515575ab-d213-42b1-aa64-ef6a2dd4641b

Inputs:

Name	Description	Type	Default Value
csource_path	Path to C program source	string	PathToAtomicsFolder/T1040/src/linux_pcapdemo.c
program_path	Path to compiled C program	string	/tmp/t1040_linux_pcapdemo

Attack Commands: Run with `bash`! Elevation Required (e.g. root or admin)

```
sudo #{program_path} -4 -P -p 17 -t 3
```

Cleanup Commands:

Preview | Code | Blame — Raw

Dependencies: Run with `bash`!

Description: compile C program

Check Prereq Commands:

```
if [ -f "#{program_path}" ]; then exit 0; else exit 1; fi
```

Get Prereq Commands:

```
cc #{csource_path} -o #{program_path}
```

Atomic Test #15 - Packet Capture Linux socket AF_PACKET,SOCK_RAW with BPF filter for UDP with sudo

Captures packets with domain=AF_PACKET,type=SOCK_RAW for a few seconds. Sets a BPF filter on the socket to filter for UDP traffic.

Supported Platforms: Linux

auto_generated_guid: b1cbdf8b-6078-48f5-a890-11ea19d7f8e9

Inputs:

Name	Description	Type	Default Value
csource_path	Path to C program source	string	PathToAtomicsFolder/T1040/src/linux_pcapdemo.c
program_path	Path to compiled C program	string	/tmp/t1040_linux_pcapdemo

Attack Commands: Run with `bash`! Elevation Required (e.g. root or admin)

```
sudo #{program_path} -a -f -t 3
```

Cleanup Commands:

```
rm -f #{program_path}
```

Dependencies: Run with `bash`!

Description: compile C program

Check Prereq Commands:

```
if [ -f "#{program_path}" ]; then exit 0; else exit 1; fi
```

Get Prereq Commands:

```
cc #{csource_path} -o #{program_path}
```

Atomic Test #16 - PowerShell Network Sniffing

PowerShell Built-in Cmdlets to capture network traffic. https://learn.microsoft.com/en-us/powershell/module/neteventpacketcapture/new-neteventsession?view=windowsserver2022-ps

Supported Platforms: Windows

auto_generated_guid: 9c15a7de-de14-46c3-bc2a-6d94130986ae

Attack Commands: Run with `powershell`! Elevation Required (e.g. root or admin)

```
New-NetEventSession -Name Capture007 -LocalFilePath "$ENV:Temp\sniff.etl"
Add-NetEventPacketCaptureProvider -SessionName Capture007 -TruncationLength 100
Start-NetEventSession -Name Capture007
Stop-NetEventSession -Name Capture007
Remove-NetEventSession -Name Capture007
```

Cleanup Commands:

```
del $ENV:Temp\sniff.etl
```