USENIX Security '25 Artifact Appendix: Haunted by Legacy: Discovering and Exploiting Vulnerable Tunnelling Hosts

Angelos Beitis
DistriNet, KU Leuven
angelos.beitis@kuleuven.be

Mathy Vanhoef
DistriNet, KU Leuven
Mathy.Vanhoef@kuleuven.be

A Artifact Appendix

A.1 Abstract

For the artifact evaluation, we provide the ZMap scanning modules, which can be used to scan the Internet to identify vulnerable tunnelling hosts. We additionally provide the scripts to test whether individual hosts are potentially vulnerable to any of the identified vulnerabilities. Lastly, we give the attack scripts for the Tunnelling-Temporal Lensing (TuTL) attack. Each of the provided tools is accompanied by a README, which further explains how the tool works and how it can be used.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

The `zmapv6-main.zip` is a software used for scanning the entire IPv4 address space, without requiring the hosting of an opt-out page or the maintenance of a blocklist. We do not recommend scanning the entire IPv4 address space for the functional evaluation; instead, you can use it to scan the individual addresses provided by us. These vulnerable hosts are under our control.

A.2.2 How to access

https://doi.org/10.5281/zenodo.15706733

A.2.3 Hardware dependencies

You will need access to a VM instance (e.g., AWS). This instance should have an IPv4 and IPv6 address which are globally routable, as well as disabled firewall rules (E.g., Your machine must be capable of sending and receiving TTL-expired packets).

A.2.4 Software dependencies

You need to have access to an Ubuntu machine which has IPv6 and IPv4 Internet access. We recommend AWS instances. Furthermore, ensure that there are no firewall restrictions potentially blocking traffic on your instance. Any additional dependencies are covered in the installation section.

A.2.5 Benchmarks

None

A.3 Set-up

A.3.1 Installation

First, create a vulnerable IP6IP6 tunnelling interface:

```
chmod +x ip6ip6.sh
sudo ./ip6ip6.sh <IPV6_ADDR> \
<IPV6_ADDR_RANDOM>
```

To install ZMap, export the ZIP file and do:

```
sudo apt-get install build-essential cmake libgmp3-
dev gengetopt libpcap-dev flex byacc libjson-c-dev
pkg-config libunistring-dev
cmake .
make -j4
sudo make install
```

For `temp_lens.zip`, export the ZIP file and do:

```
sudo apt install tcpreplay
pip install netifaces
pip install scapy
```

For `tunneltester.zip`:

```
python3 -m venv venv
source venv/bin/activate
pip install wheel scapy==2.6.1
sudo su
source venv/bin/activate
```

A.3.2 Basic Test

Tunneltester: For tunneltester

```
sudo python3 tunnel_tester.py \
<INTERFACE> -t6 \
2600:1f18:2869:9b00:5b7b:92c9:d861:d151 \
-P6 <IPv6>
```

ZMap: For ZMap:

```
sudo zmap 34.171.178.123 -M ipip \
- -external-ipv4-address <IPv4_ADDRESS>
```

A.4 Evaluation workflow

A.4.1 Major Claims

Our artifacts provide the means to identify vulnerable hosts, either individual machines or multiple machines, through Internet-wide scans.

(C1): Our artifacts can successfully identify vulnerable tunnelling hosts.

(C2): Our attack script can successfully be used to conduct a tunnelled-temporal lensing attack as described in the paper. For the functional badge, the tunneled-temporal lensing script needs to be able to collect latencies and send packets that will eventually reach a host.

A.4.2 Experiments

Instructions on how to run each component.

(E1): *[Tunneltester] [5 human-minutes + 2 compute-minutes]:*
How to: The tunneltester script will detect two of our own hosts for the listed vulnerabilities in the paper. Replace <INTERFACE> with your machine's main interface, e.g., eth0, <PRIVATE_IPV4> with the private IPv4 address of your machine in case it is behind a NAT (can be left empty if this is not the case), and <PUBLIC_IPV4> and <IPv6> with your VM's global IPv4 and IPv6 addresses, respectively.
Execution: Run the following two commands:

```
sudo python3 tunnel_tester.py \
<INTERFACE> -t 3.90.110.118 -t6 \
2600:1f18:2869:9b00:5b7b:92c9:d861:d151 \
-s 172.31.29.15 -s6 \
2600:1f18:2869:9b00:5b7b:92c9:d861:d151 \
-p <PRIVATE_IPv4> -P <PUBLIC_IPv4> \
-P6 <IPv6>
```

```
sudo python3 tunnel_tester.py <INTERFACE> \
-t 34.171.178.123 \
-s 34.171.178.123 \
-p <PRIVATE_IPv4> -P <PUBLIC_IPv4>
```

Results: The script will output whether or not the host is vulnerable to a specific protocol. Specifically, you are looking to see the `VULNERABLE` text. The 3.90.110.118 host should be vulnerable to IPv6-based protocols (such as IP6IP6, GRE6 etc). AWS cannot send TTL-expired packets, meaning TTL-expired scans will not detect vulnerable hosts. The 34.171.178.123 host is vulnerable to IPv4-based protocols, but Google Cloud blocks the host from replying to ICMP Echo requests.

(E2): *[ZMap Scanning modules] [5 human-minutes + 2 compute-minutes]:*
How to: The ZMap Scanning module is used to identify vulnerable tunnelling hosts on the Internet. It can also be used to scan for an individual host. We provide a script named `zmap.sh`, which will go over some of the scanning modules to scan specific hosts under our control.
Execution: Run the following command:

```
chmod +x zmap.sh
sudo ./zmap.sh <YOUR_IPv4_ADDRESS>
<YOUR_IPv6_ADDRESS>
```

Results: The script goes over the scanning modules. You should see for each one the following:
`1 done (137 p/s avg); recv: 1 1 p/s`, indicating that the host scanned has been identified in the scan. For the final 4in6 scan, the scanning module will not detect the host since the inner source IP will be translated and the host does not spoof. To circumvent this, the script will catch the traffic in tcpdump. You should see the following:
`IP 3.90.110.118 > <YOUR_IPv4>: ICMP echo reply...`

(E3): *[Tunneled-Temporal Lensing Attack] [15 human-minutes + 10 compute-minutes]:*
How to: The tunnelled-temporal lensing attack can be used to concentrate traffic towards the attacker in time. For convenience, this script assumes the target is also a vulnerable tunnelling host. You will need to first create a vulnerable IP6IP6 interface, by running the script `ip6ip6.sh` as explained in Section A.3.
Execution: First create a vulnerable tunnelling interface by running:

```
sudo ./ip6ip6.sh <IPV6_ADDR> \
<IPV6_ADDR_RANDOM>
```

Where <IPV6_ADDR_RANDOM> is the IPv6 address for the newly created interface.
Prepare two new screens to capture the traffic:

```
screen -S outgoing
Press Ctrl+AD
screen -S incoming
Press Ctrl+AD
screen -S attack
Press Ctrl+AD
```

then create a file named `spoofers.txt` and add the two addresses listed below inside:

```
chmod +x create_spoofers.sh
sudo ./create_spoofers.sh \
2600:1f18:2869:9b00:bec4:73f4:9a4e:18ce \
2600:1f18:2869:9b00:21b7:96f:47b9:eed7
```

Run the following script. We set the sample size to 10. In the paper, the sample was 1000, but for convenience, we will use a short sample size of latencies in the artifact functionality assessment:

```
screen -r attack
sudo python3 -m src.main -i eth0 -vic
<IPV6_ADDRESS> -type ipip6 -alg bf -att 6
-sample 10 -spoo_f ./spoofers.txt
```

You should start seeing `Message Received!`. After latency collection is complete, you will be asked to start the attack. `Ready to attack? yes/no`: Do not press anything yet. Then do:

```
Press Ctrl+AD
screen -r outgoing
sudo ./outgoing.sh <IPv6 Address>
Press Ctrl+AD
screen -r incoming
sudo ./incoming.sh <IPv6 Address>
Press Ctrl+AD
screen -r attack
```

Now you will be back where you are asked to `Ready to attack? yes/no:`. Type yes and press enter. You can then press no to the following prompt.
The two files `outgoing.pcap` and `incoming.pcap` will be created after terminating the two scripts. These files have the incoming and outgoing traffic of the attack, respectively. This traffic can then be used for analysis in `wireshark`. Specifically, when opened in `wireshark`, you can go to `Statistics > I/O Graphs` and set the interval to 20ms.

A.5 Notes on Reusability

The addresses provided here are vulnerable and under our control and will be shut down after the artifact evaluation phase to prevent abuse.

A.6 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2025/.