ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

USENIX'23 Artifact Appendix: Device Tracking via Linux's New TCP Source Port Selection Algorithm

Moshe Kol
Hebrew University of Jerusalem

Amit Klein
Hebrew University of Jerusalem

Yossi Gilad
Hebrew University of Jerusalem

A Artifact Appendix

A.1 Abstract

This artifact contains a proof-of-concept implementation of a device tracking technique for Linux-based devices by exploiting the way Linux selects TCP source ports. The Linux TCP port selection algorithm is an adaptation of Algorithm 4 ("Double-Hash Port Selection Algorithm") from RFC 6056. The algorithm is used starting from kernel version `5.12-rc1`.

The artifact contains a tracking server written in Go and a tracking snippet written in HTML+JavaScript, served by the tracking server using HTTP.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

Our attack generates a device ID for the tested device, that can identify it across browsers, browser privacy modes, networks, containers and VPNs. Therefore, for the sake of maintaining evaluator's privacy, we recommend evaluating our artifact on a local/private network.

In addition, in the "Set-up" section, we instruct the evaluator to use older versions of Ubuntu 22.04 and Google Chrome. Older versions are at risk of security bugs, therefore using local network is preferred.

A.2.2 How to access

The artifact is available on GitHub:
`https://github.com/0xkol/rfc6056-device-tracker/tree/09dd6ab68e10566eb6ca7760ef78d4689c7e2b85`

A.2.3 Hardware dependencies

8GB of RAM, 4 CPU cores and 50GB free disk space.

A.2.4 Software dependencies

Tracking client requirements

1. **Linux kernel:** The Linux kernel of the client device must be one of the following versions: 5.12.*, 5.13.*, 5.14.*, 5.15–5.15.40, 5.16.*, 5.17–5.17.8.
2. **Google Chrome:** version 96.0.
3. (Optional) **Python:** `Python 3.5` or above.

Tracking server requirements We assume that the tracking server runs on a Linux host.

1. `Go` version `1.18`, the `google/gopacket` library and the `google/gopacket/pcap` library.
2. `libpcap-dev` package (on Ubuntu).
3. `git`.

A.2.5 Benchmarks

None.

A.3 Set-up

A.3.1 Installation

To evaluate our attack, you will need two Linux machines: one for our tracking server and one for the tracking client (that interacts with the server using Chrome). The client machine has specific Linux kernel constraint, so we recommend using a virtual machine (VM) for it. The tracking server can run on any Linux machine that has network connectivity (IPv4 and IPv6) to the client machine. In this document we describe how to run both server and client as (separate) virtual machines.

Configure Oracle VirtualBox: Download and Install Oracle VirtualBox from this URL `https://www.virtualbox.org/wiki/Downloads`.

Configure Host-Only Network on VirtualBox:

1. **Disable address range control (required on Linux hosts only):** Create the file `/etc/vbox/networks.conf` and write this line to it (including the asterisk): `* 0.0.0.0/0 ::/0`

2. **Open "Host Network Manager":** Open VirtualBox and click on "File" → "Host Network Manager".

3. **Create a New Host-Only Network:** On "Host Network Manager", click on the "Create" button. This will create a new interface on your host with the name `vboxnet0` (or similar).

4. **Configure IPv6:** By default, only IPv4 prefix will be assigned to the new virtual interface (`192.168.56.1/24` or similar). To configure IPv6, use our pre-generated ULA prefix `fd3f:e8d8:3a1f:0::/64`. On the "IPv6 address" field enter `fd3f:e8d8:3a1f:0::1` and on the "IPv6 Prefix Length" enter `64`.

5. Click on the "Apply" button. You should see no errors.

Tracking Client Installation: We describe here how to setup a tracking client machine using Oracle VirtualBox.

1. **Download and Install Ubuntu 22.04:** Download Ubuntu Desktop 22.04 (not 22.04.1) from this URL `http://old-releases.ubuntu.com/releases/jammy/ubuntu-22.04-desktop-amd64.iso`. Install Ubuntu as a new Virtual Machine on Oracle VirtualBox. Notes:

   - You may follow these instructions for reference: `https://brb.nci.nih.gov/seqtools/installUbuntu.html`
   - We recommend assigning to the VM 4GB of RAM, 2 CPUs and 20GB of disk space.
   - **Avoid downloading updates during installation.** Otherwise, Ubuntu will auto-update its kernel. Also, make sure your machine is **NOT connected to the Internet** during installation by changing the network adapter from "NAT" to "Not attached" in the VM "Settings" window.
   - The kernel version of your installed Ubuntu should be `5.15.0-25-generic`. You can view it using the command: `uname -a`

2. **Connect to the Internet:** When your VM is up and running, connect it to the Internet by changing the network adapter from "Not attached" to "NAT" in the VM "Settings" window. (Avoid updating Ubuntu if it prompts for an update.)

3. **Download and Install Google Chrome:** Download Google Chrome v96.0 from `https://dl.google.com/linux/chrome/deb/pool/main/g/google-chrome-stable/google-chrome-stable_96.0.4664.110-1_amd64.deb`

   Install using the following command: `sudo dpkg -i google-chrome-stable_96.*.deb`

4. **Switch to Host-Only Network:** Open the VM "Settings" window. On the "Network" tab change the network adapter to "Host-only adapter" and choose the name of the adapter you created previously (probably `vboxnet0` or similar).

5. **Configure IPv6:** Use the following command to ensure IPv6 connectivity between the VMs:

   ```
   sudo ip address add
       fd3f:e8d8:3a1f:0::10/64 dev IFNAME
   ```

   To find *`IFNAME`*, list the network adapters on the machine using the `ip address` command and note the interface name whose name is not `lo`. Beware: This command does not survive reboot.

6. (Optional) You can verify that the machine you installed is vulnerable (i.e. uses the un-patched version of Algorithm 4 of RFC 6056) by invoking our Python 3 detection tool: `python3 CVE-2022-32296_tester.py`

   Expected output:

   ```
   Verdict: RFC 6056 Algorithm 4 (Vulnerable)
   ```

Tracking Server Installation:

1. Install Ubuntu Desktop 22.04 on a separate virtual machine, similar to the "Tracking Client Installation".

2. **Install Packages:** Run the following commands:

   ```
   sudo apt update
   sudo apt install git golang-go libpcap-dev
   ```

3. **Clone Repository:** Clone the git repository using the following commands:

   ```
   git clone
       https://github.com/0xkol/rfc6056-device-
       tracker.git
   cd rfc6056-device-tracker
   git checkout 09dd6ab
   ```

4. **Install Go Libraries:** On the repository folder, type the following commands:

   ```
   go get github.com/google/gopacket
   go get github.com/google/gopacket/pcap
   ```

5. **Switch to Host-Only Network:** Similar to what you did for the client machine, change the network adapter to "Host-only adapter" and choose the name of the adapter you created previously. After this step, both the client and server VMs should be up and running, with their network adapter configured to the same Host-Only network created previously.

6. **Configure IPv6:** Similar to what you did on the client machine, type the following command:

   ```
   sudo ip address add
       fd3f:e8d8:3a1f:0::20/64 dev IFNAME
   ```

   Beware: This command does not survive reboot.

7. **Compile and Run the Tracking Server:** Switch to the git repository folder. Then, compile the tracker using: `go build -o tracker tracker.go`

   The compilation should succeed (no output on the console). Proceed by running the server on the interface you discovered on the previous step. For example (assuming the interface is `enp0s3`): `sudo ./tracker -iface enp0s3`

   You should see the output:

   ```
   RFC 6056 Device Tracker v1.3 start
       (capturing on: enp0s3)
   ```

A.3.2 Basic Test

Connectivity Test: By now you should have two VMs connected to the same Host-Only network, with IPv4 and IPv6 connectivity. Verify that you can ping from the client VM to the server VM by issuing:

```
ping6 fd3f:e8d8:3a1f:0::20
ping SERVER_IPV4_ADDRESS
```

You can find *`SERVER_IPV4_ADDRESS`* by issuing the `ip address` command on the server machine.

Browser Test: On the client VM, open the Google Chrome browser and browse to the server using both IPv4 and IPv6 (i.e. to URLs `http://[fd3f:e8d8:3a1f:0::20]/` and `http://`*`SERVER_IPV4_ADDRESS`*`/`). You should see a webpage with the title "RFC 6056 Device Tracker Demo".

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): The same device ID is obtained in a Google Chrome regular tab and an incognito tab. Section 6.1 in the paper. Proven by experiment (E1).

(C2): The same device ID is obtained when using IPv4/IPv6. Section 6.2 in the paper. Proven by experiment (E2).

(C3): The dwell time on Google Chrome varies between 5-15 seconds, depending on the RTT to the tracking server and the physical machine. Section 6.4 in the paper. Proven by experiment (E3).

A.4.2 Experiments

In all of the experiments, you should verify that the tracking server is up and running, and that it has both IPv4 and IPv6 connectivity from the tracking client.

(E1): Cross browser privacy modes consistency. 30 human-minutes, 30 compute-minutes.
Tracking Client VM Preparation: Open two Google Chrome windows: a regular window and an incognito window. On each window, browse to the tracking server. *Make sure that the "Tracker address" field contains the IP address of the tracking server.*
Execution: On the normal window, hit "Fingerprint me!" to launch the fingerprinting process. Few seconds later, you should see "fingerprint" and "fingerprint hash" on the webpage. Write these down for later. Continue by hitting "Fingerprint me!" on the incognito window and ensure you get the same fingerprint. Avoid running two fingerprinting measurements simultaneously.
Results: The same fingerprint should be generated on each window.

(E2): Cross protocol consistency. 30 human-minutes, 30 compute-minutes.
Tracking Client VM Preparation: Open a Google Chrome window (normal one is enough), and browse to the tracking server (over IPv4 or IPv6, does not matter).
Execution: Fingerprint the client machine once using an IPv4 address of the tracking server. Write down the fingerprint for a later comparison. Fingerprint the client machine again, but now use IPv6 as the tracking server. (On the "Tracker address" field use `[fd3f:e8d8:3a1f:0::20]` (including brackets!).) Verify that you get the same fingerprint on each run.
Results: The same fingerprint should be generated for both IPv4 and IPv6.

(E3): Dwell time. 30 human-minutes, 30 compute-minutes.
Tracking Client VM Preparation: Open a Google Chrome window and browse to the tracking server.
Execution: Fingerprint the client machine using IPv4 or IPv6 (it doesn't matter which at this point) and write down the "total time" reported in the webpage. Repeat the experiment a few times to obtain an average readout.
Results: You should observe an average dwell time of 5-15 seconds, depending on the network RTT and physical machine.

A.5 Version

Based on the LaTeX template for Artifact Evaluation V20220912. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at `https://secartifacts.github.io/usenixsec2023/`.