USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Catch-22: Uncovering Compromised Hosts using SSH Public Keys

Cristian Munteanu, *Max Planck Institute for Informatics;*
Georgios Smaragdakis, *Delft University of Technology;*
Anja Feldmann and Tobias Fiebig, *Max Planck Institute for Informatics*

https://www.usenix.org/conference/usenixsecurity25/presentation/munteanu

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 34th USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 34th USENIX Security Symposium.

August 13–15, 2025 • Seattle, WA, USA

978-1-939133-52-6

Open access to the Artifact Appendices to the Proceedings of the 34th USENIX Security Symposium is sponsored by USENIX.

USENIX Security '25 Artifact Appendix: Catch-22: Uncovering Compromised Hosts using SSH Public Keys

Cristian Munteanu
Max Planck Institute for Informatics

Georgios Smaragdakis
Delft University of Technology

Anja Feldmann
Max Planck Institute for Informatics

Tobias Fiebig
Max Planck Institute for Informatics

A Artifact Appendix

A.1 Abstract

This paper presents an Internet-scale scanning tool designed to identify compromised SSH servers by analyzing SSH behavior during public key authentication. Our tool detects malicious SSH keys by sending authentication challenges without accessing the compromised systems or requiring audit-level privileges. This approach ensures a non-intrusive and ethical method for assessing exploited SSH servers.

A.2 Description & Requirements

This section describes our scanning tool, including hardware and system requirements, as well as ethical considerations. Additionally, we provide installation and testing instructions.

A.2.1 Security, Privacy, and Ethical Considerations

Our study uses a set of public keys known to be involved in malicious activities. These keys were shared with us through our collaboration with Bitdefender and cannot be publicly disclosed. Revealing these keys would diminish the effectiveness of our scans, as malicious actors could identify and replace their compromised keys. However, some of these keys are publicly available and can be found through a simple web search. For Experiment 1 A.4.2, we have additional resources, including IP prefixes, IP addresses, and SSH keys, which cannot be publicly disclosed. These materials will be provided to the evaluation committee during the evaluation process.

On another note, the nature of the artifact must be carefully considered. The scanning tool, particularly when used at a large scale, can place significant pressure on the scanned networks. This may lead to unintended consequences, including unforeseen disruptions and dissatisfied system administrators. To mitigate these risks, it is essential to maintain a blocklist, ensure 24/7 contact availability, and carefully regulate the scan rate. Additionally, for an Internet-wide scan, obtaining approval from an Institutional Review Board (IRB) is mandatory.

A.2.2 Accessing the Artifact

The artifact is available at: https://edmond.mpg.de/dataset.xhtml?persistentId=doi%3A10.17617%2F3.LVPCS6

A.2.3 Hardware Requirements

The recommended *minimum* system specifications:
- 4 CPU cores
- 4 GB of RAM
- An active Internet connection

For a *full-scale Internet-wide scan*, we recommend using at least four machines, each with the following specifications:
- 16 core AMD EPYC 7232 or better
- 64 GB memory
- 10gbit network connection
- at least 4TB NVMe in raid 1

A.2.4 Software Requirements

Our scanning tool was tested and deployed on Debian 12. It requires ZMap [1] for port scanning and a patched version of ZGrab2 (included in the artifact). The scanning script will automatically download ZMap and build the patched ZGrab2.

The following software dependencies are required:

- Go (version 1.19 or later) – Install using: `sudo apt-get install golang` (Debian 12);
- Bash (`/bin/bash`) – Installed by default on a standard Debian 12 system.

A.2.5 Benchmarks

None.

[1] https://github.com/tumi8/zmap

A.3 Set-Up

Follow these steps to set up the scanning tool:

- Download the *scanner.tar.gz*
- Extract the tarball using: `tar -xvf scanner.tar.gz`.
- Navigate to the "scanner" directory. This directory contains the `scan_ssh_ip.sh` script and the `Zgrab2_patched` folder, which holds the patched ZGrab2 source code.

The contents of the "scanner" directory should look like this:

```
$ ls -alh scanner
drwxr-xr-x .
drwxr-xr-x ..
drwxr-xr-x Zgrab2_patched
-rwxr-xr-x scan_ssh_ip.sh
```

A.3.1 Installation

The primary dependency is Go. To install Go on Debian 12, run:

```
sudo apt-get install golang
```

A.3.2 Basic Functionality Test

Ensure you are in the "scanner" directory.

Execute:

```
./scan_ssh_ip.sh -help
```

Expected output:

```
Go is installed.
go version go<version>
Usage: ./scan_ssh_ip.sh [OPTIONS]

Options:
...
```

A.4 Evaluation Workflow

This section outlines the necessary steps to evaluate the functionality and reproducibility of our artifact.

A.4.1 Major Claims

The primary contributions of our tool are:

(C1): The scanning tool offers a reliable solution for determining whether a specific SSH public key is installed for a particular user on a publicly accessible SSH server.

(C2): Using this tool in a global scan, we detected over 21,700 unique compromised systems spanning 1,649 ASes and 144 countries, affected by 52 verified malicious keys provided by a threat intelligence partner. The scan uncovered critical Internet infrastructure vulnerabilities, highlighted campaigns such as the 'fritzfrog' IoT botnet and actors like 'teamtnt', and identified state-associated malicious keys within sensitive networks. By correlating findings with honeypot data, we revealed disparities in attacker activity representation, including underestimated APT operations.

A.4.2 Experiments

Experiment 1: Controlled environment This experiment will validate our first claim (C1) regarding the reliability of the SSH public key scanning solution. For the experiment, we deployed multiple SSH servers preconfigured with specific SSH keys. These test servers will be accessible only during the artifact evaluation. Additional details, including IP prefixes, SSH public/private keys, and configuration files, will be provided to the evaluation committee (see Section A.2.1).

Each test system is configured with various user accounts ('root', 'test', 'foo', and 'bar') and a predefined set of SSH keys. To facilitate testing, we provide:

- A list of IPv4 prefixes and IPv6 addresses: `ipv4_prefix_list` and `ipv6_list`;
- A collection of SSH public keys for testing: `mal_k_round_0`, `mal_k_round_1`, `mal_k_round_2`;
- A "ground-truth" file to compare experimental results: `sample_output.csv`;
- A shell script to streamline the experiment (Figure 1): `demo_run.sh`;
- A set of private SSH keys for non-root users to validate the scanner's capability. If the scanner detects a public key for a user on an SSH server, the corresponding private key can be used to establish a connection. Note that 'root' user keys are not shared due to security concerns: `demo_keys/`.

To conduct the experiment, you will need a machine–either a physical system or a virtual machine (VM)–that meets the recommended *minimum* system specifications (see Section A.2.3) and runs Debian 12. Follow these steps to run the experiment:

(Step 1) Place all additional resources, provided for the evaluation, in the same directory as the scanner script ("scanner").

(Step 2) Ensure the script "demo.sh" is executable: `chmod +x demo.sh`

(Step 3) Run the script: `./demo.sh`

After adding the additional resources, the contents of the "scanner" directory should look like this:

```
$ ls -alh scanner
drwxr-xr-x .
drwxr-xr-x ..
drwxr-xr-x demo_keys
drwxr-xr-x Zgrab2_patched
-rwxr-xr-x demo_run.sh
-rw-r--r-- ipv4_prefix_list
-rw-r--r-- ipv6_list
-rw-r--r-- mal_k_round_0
-rw-r--r-- mal_k_round_1
-rw-r--r-- mal_k_round_2
-rwxr-xr-x scan_ssh_ip.sh
```

After the script execution, the results can be found the `demo_run.csv` file. The Zmap and Zgrab2 output files can be found in the `ssh_scan_results_v4/` and `ssh_scan_results_v6/` folders.

Experiment 2: Internet-Compromised SSH This experiment will validate our second claim (C2) regarding the feasibility of conducting a full-scale Internet scan and accurately identifying SSH servers compromised by known malicious SSH keys. The data for our study was generated through this experiment.

For a *full-scale Internet-wide scan*, we recommend using at least four machines, each meeting the specifications outlined in Section A.2.3. We strongly advise using physical machines rather than virtual machines (VMs) to prevent potential network-related issues. A VM may struggle to handle a sustained high volume of packets, potentially biasing the results.

To replicate this experiment, we define a series of steps. However, some of these steps are not strictly deterministic and cannot be precisely reproduced; they should be considered as guidelines. Nevertheless, each step is crucial and must be executed.

It is important to note that certain steps may take an unpredictable amount of time, affecting the overall duration of the experiment. For example, the Institutional Review Board (IRB) approval process can take anywhere from two weeks to six months, depending on the IRB's policies and workload. Additionally, the IRB may impose limitations on the scanning speed (e.g., restricting scans to no more than 50,000 packets per second).

As a benchmark, in our experiment, obtaining ethical approval took over three weeks. Conducting a full IPv4 scan for a single user with 52 keys required approximately 13 days, followed by an additional 3-4 days to scan the IPv6 addresses in the `ipv6hitlist`.

Follow these steps to run the experiment:

(Step 1) Obtain SSH public keys. Since the public keys we used cannot be shared (see Section A.2.1), you can instead source malicious SSH public keys from publicly available locations such as security blogs or public databases of Indicators of Compromise (IOCs). For instance, a widely known malicious SSH public key can be identified by its label–'mdrfckr'–which is referenced in our paper as `MK06`.

(Step 2) Define the target IP address range. For IPv4 scanning, you can use the '0.0.0.0/0' prefix to cover the entire IPv4 address space. For IPv6, consider using the `ipv6hitlist` service to obtain a list of active IPv6 addresses.

(Step 3) Build an IP blocklist for IPv4 and IPv6.

- For IPv4, a good starting point is the IANA IPv4 Special-Purpose Address Registry, which contains non-routable and reserved addresses.
- Store the blocklist prefixes in a separate file and specify it when running the scan script.
- For IPv6, since entire prefixes cannot be scanned, filtering must be done at the individual IP level. Begin by removing IPs listed in the IANA IPv6 Special-Purpose Address Registry. Tools such as `grepcidr` can assist in this process.

(Step 4) Obtain Institutional Review Board (IRB) approval. For large-scale scans, obtaining IRB approval is mandatory before execution. The IRB committee will assess the ethical implications of the experiment and determine the appropriate scanning rate while ensuring compliance with best practices (Menlo Report [2], Drumenic et al.). Setting up informative rDNS, maintaining accurate WHOIS records, hosting a website that explains your scanning activities, and ensuring 24/7 availability via phone and email would constitute the requirements for adhering to best practices.

(Step 5) Run the scan. Start the scan using the script provided in the artifact. Throughout the scanning process, ensure that the blocklist remains up to date to prevent unintended scanning of restricted or sensitive IP addresses.

A.5 Notes on Reusability

This artifact is designed for long-term reusability. It provides the necessary tools to detect SSH-compromised hosts across the Internet. Anyone collecting malicious SSH public keys can use this artifact to identify systems compromised by those keys.

A comprehensive 'README' file is included in the artifact, detailing:

[2] https://www.dhs.gov/sites/default/files/publications/CSD-MenloPrinciplesCORE-20120803_1.pdf

```
#!/bin/bash

OUT_FILE="demo_run.csv"

# Check if jq is installed
if command -v jq &> /dev/null; then
    echo "JQ is installed."
else
    echo "Installing JQ..."
    sudo apt-get update && sudo apt-get -y install jq
fi
jq --version

echo "====== Starting Demo ======"
echo "Estimated run time per user: ~3m40s"
echo "Total estimated time: ~30 min"

# Run scans for different users over IPv4
for user in foo bar test root; do
    ./scan_ssh_ip.sh -ipwhitelist=ipv4_prefix_list -port=22 -user=$user -scantype=4
       -wait=10
done

# Run scans for different users over IPv6
for user in foo bar test root; do
    ./scan_ssh_ip.sh -ipset=ipv6_list -port=22 -user=$user -scantype=6 -wait=10
done

echo "IP,USER,Key" > $OUT_FILE
cat ssh_scan_results*/*/key_* | grep "key-accepted\":true" | jq -r '.ip + ",
   " + .data.ssh.result.dummyauth.user + ",
   " + .data.ssh.result.dummyauth.key' >>
   $OUT_FILE
echo "Demo run finished. Output file: $OUT_FILE"
```

Figure 1: SSH script for demo run.

- How to execute a scan
- How to adjust scan parameters (e.g., wait times, network interface selection)

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/.