ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE
ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL
ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

USENIX Security '24 Artifact Appendix: Terrapin Attack: Breaking SSH Channel Integrity By Sequence Number Manipulation

Fabian Bäumer
Ruhr University Bochum

Marcus Brinkmann
Ruhr University Bochum

Jörg Schwenk
Ruhr University Bochum

A Artifact Appendix

A.1 Abstract

This document describes the artifacts to the USENIX Security '24 Publication *Terrapin Attack: Breaking SSH Channel Integrity By Sequence Number Manipulation*.

Using these instructions, the evaluations of Sequence Number Manipulation (Sect. 4.1), Extension Downgrade Attack (Sect. 5.2), Rogue Extension Attack (Sect. 6.1) and Rogue Session Attack (Sect. 6.2) can be reproduced.

Also, the aggregation scripts for the internet scans are available and can be tested on a small subset of the samples.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

The configuration uses the host network to allow (optional) monitoring of the attack using Wireshark or other network packet analysis tools on the loopback interface. During the runtime of the evaluation, this makes the tested SSH server and proof of concept (PoC) available to all systems with access to the local network (TCP bind to `0.0.0.0`, ports 2200 and 2201). Reviewers should take care to isolate the test system from the internet, for example using a firewall.

A.2.2 How to access

The artifacts are publically available at `https://github.com/RUB-NDS/Terrapin-Artifacts/tree/9907c80fa7e4184a29ceac352947ea51a49dce6a`.

A.2.3 Hardware dependencies

None.

A.2.4 Software dependencies

- Linux or MacOS.[1] No specific distribution or version is required. We used Manjaro (rolling release in March 2024) and MacOS 14.4 (Sonoma).
- Bash shell interpreter (typically included in the above). No specific version is required. We used bash 5.2.26 and 3.2.57.
- Docker Engine or Docker Desktop. While Docker Engine suffices and is typically included in Linux distributions, Docker Desktop is a separate install on MacOS. No specific version is required. We used Docker Engine 25.0.3 and Docker Desktop 4.28.0.
- Wireshark (optional), for network packet analysis.

A.2.5 Benchmarks

None.

A.3 Set-up

All required Docker images are built on demand when the evaluation scripts are executed, so no setup is required.

The TCP ports 2200 and 2201 should be free and available. This is the case by default on many systems. Some systems might require a configuration of the firewall to allow the test servers to bind `0.0.0.0` on these ports. On some systems, the firewall will show a pop-up dialog when the first server starts up, requiring manual confirmation.

A.3.1 Installation

Linux: Install the Docker engine under a supported Linux distribution by following the instructions available at `https://docs.docker.com/engine/install/`.

MacOS: Install Docker Desktop available at `https://www.docker.com/products/docker-desktop/`.

A.3.2 Basic Test

The following scripts build all required Docker images and can be used as a basic functionality test. It will also be called by all evaluation scripts, so this step is optional.

```
$ impl/build.sh
[+] Building 2.0s (15/15) FINISHED
[...]
=> => naming to docker.io/terrapin-artifacts/openssh-
    ↪ server:9.4p1
```

[1] Windows WSL might work but is untested and *not* supported.

Attack	PuTTY 0.79	OpenSSH 9.4p1	OpenSSH 9.5p1	Dropbear 2022.83	AsyncSSH 2.13.2	libssh 0.10.5
C1 RcvIncrease	✓	-	✓	✓	✓	✓
C1 RcvDecrease	✓	-	R	✓	T	T
C1 SndIncrease	✓	-	R	U	T	T
C1 SndDecrease	✓	-	R	U	T	T
C2 ChaCha-Poly	✓	✓	✓	-	-	-
C2 CBC-EtM						
- UNKNOWN	0.0300	0.0003	0.0003	-	-	-
- PING	0.8383	-	0.0074	-	-	-
C3 Rogue Extension	-	-	-	-	✓	-
C4 Rogue Session	-	-	-	-	✓	-

$^{-}$ Not evaluated.
$^{\checkmark}$ Attack succeeds.
^{R} Client terminates the connection (rollover).
^{T} Client terminates the connection (timeout).
^{U} Client terminates the connection (UNKNOWN message).

Table 1: Expected outcomes for attacks against clients

```
[...]
$ pocs/build.sh
[...]
```

The output shows the progress on downloading base images and building the evaluation images. If there is no output, all docker images are already built.

A.4 Evaluation workflow

A.4.1 Major Claims

We evaluated our attacks against several clients using an OpenSSH 9.5p1 (C1, C2) or AsyncSSH 2.13.2 (C3, C4) server. For an overview of the expected outcomes, see also Table 1.

(C1): Sequence Number Manipulation (Sect. 4.1). *We verified all techniques successfully against PuTTY 0.79. Additionally, our experiments show that OpenSSH 9.5p1 recognizes a rollover of sequence numbers and terminates the connection, thus not affected by any technique but RcvIncrease. AsyncSSH 2.13.2 and libssh 0.10.5 allow for RcvIncrease but terminate the connection due to handshake timeouts before any advanced technique concludes. Dropbear 2022.83 disconnects on* UNKNOWN *messages instead of responding with* UNIMPLEMENTED *but allows* Rcv *to roll over, therefore being affected by RcvIncrease and RcvDecrease only.*

(C2): Extension Downgrade (Sect. 5.2). *We successfully evaluated the attack in 10,000 trials on ChaCha20-Poly1305 and CBC-EtM against OpenSSH 9.5p1 and PuTTY 0.79 clients, connecting to OpenSSH 9.4p1 (*UNKNOWN *only) and 9.5p1. For CBC-EtM, our success rate in practice was 0.0003 (OpenSSH) resp. 0.0300 (PuTTY), improved to 0.0074 (OpenSSH) resp. 0.8383 (PuTTY) when sending* PING *instead of* UNKNOWN.

Shortcut	Description
q	Quit
h	Help
S	Wrap long lines on/off
/	Search
:n	Next file
:p	Previous file

Table 2: Common keyboard shortcuts of `less`.

(C3): Rogue Extension Negotiation (Sect. 6.1). *We successfully evaluated the attack against AsyncSSH 2.13.2 as a client, connecting to AsyncSSH 2.13.2.*

(C4): Rogue Session Attack (Sect. 6.2). *We successfully evaluated the attack against AsyncSSH 2.13.2 as a server, connecting to AsyncSSH 2.13.2.*

Internet Scan (Sect. 7) We are also including sample data, aggregated data, and evaluation scripts on the Internet scan.

A.4.2 Experiments

The evaluation scripts (in the directory `scripts`) are interactive and self-describing. Some of them have several output files. In that case, the files (as described below) are all opened in the text file viewer `less` *at the same time*, requiring keyboard-based navigation to see all of the results. As a gentle introduction to `less`, see Table 2 for a quick reference of useful keyboard shortcuts.

(E1): `test-sqn-manipulation.sh` [$\approx 1-3$ hours per client/variant combination]: Run one of the four sequence number manipulation attacks to prove **(C1)**. RcvIncrease is very fast; the others can be slow.
Execution: After starting the script, choose a client, one of the four attack options, and input the manipulation offset N. To prove (C1), input $N = 1$.
Results: The attack is complete once the progress bar fills. After that, there will be an error message because the secure channel is broken, as the script does not implement any prefix truncation to complete the attack.

(E2a): `test-ext-downgrade.sh` [$\approx$ 1 minute]: Run the extension downgrade attack to prove **(C2) for ChaCha20-Poly1305**.
Execution: After starting the script, choose an arbitrary client and server combination. Afterward, choose attack variant `1` to select ChaCha20-Poly1305.
Results: The script will conclude by opening the following files simultaneously in `less`:

1. `diff` of files 3 and 4

2. `diff` of files 5 and 6
3. Server log (unmodified connection)
4. Server log (tampered connection)
5. Client log (unmodified connection)
6. Client log (tampered connection)
7. PoC proxy log

Navigate to the second file. The file compares the output of the selected SSH client in the case of an extension downgrade attack to the output of an unmodified connection. The diff will indicate the presence of `SSH_MSG_EXT_INFO` and absence of `SSH_MSG_IGNORE` in the unmodified connection only, thus proving (C2) for ChaCha20-Poly1305.

(E2b): `bench-ext-downgrade.sh` [≈ 1 − 2 hours per client/variant combination]: Run the extension downgrade attack 10,000 times to prove **(C2) for CBC-EtM (Unknown and Ping)**.
Execution: After starting the script, choose between Unknown and Ping variants of the attack, then select between OpenSSH and PuTTY client. A progress bar will show the current trial.
Results: After finishing all trial connections, the number of successful trial runs will be outputted to the console. The relative success rate will be close to the values claimed in (C2), thus proving the functionality and success probability claims in (C2) in the case of CBC-EtM.

(E3): `test-asyncssh-rogue-ext-negotiation.sh` [≈ 1 minute]: Run rogue extension attack to prove **(C3)**.
Execution: The attack is automatic.
Results: The script will conclude by opening a set of seven files in `less`. Refer to the results of (E2a) for a list of files opened. Navigate to the second file. The diff will indicate the presence of the `server-sig-algs` extension with an attacker-chosen value in the tampered connection, thus proving (C3).

(E4): `test-asyncssh-rogue-session-attack.sh` [≈ 1 minute]: Run the rogue session attack to prove **(C4)**.
Execution: The attack is automatic.
Results: The script will conclude by opening a set of seven files in `less`. Refer to the results of (E2a) for a list of files opened. Navigate to the first file. The diff will indicate successful authentication for the victim (unmodified connection) and attacker (tampered connection), respectively. Afterward, navigate to the second file and examine the output of each client connection at the end of the file. In the unmodified connection, the server will respond with the username victim, while in the attacked connection, the server will respond with the username attacker. This proves (C4).

(E5): `scan_util.py` [≈ 1 minute]: Run the script to aggregate a set of zgrab2 scan results. Note that this script is in the sub-directory `scans`. Without the full data, we can not prove the statistics in our paper. However, we can demonstrate how we aggregated the scan results and classified the algorithms.
Execution: Build the docker image by running the following command inside the `scans` directory:

```
$ docker build . -t terrapin-artifacts/scan-util
```

Now aggregate the `sample.json` file which can be found in the `sample` sub-directory by running the following command inside the `scans` directory:

```
$ docker run --rm -v ./sample:/input terrapin-
    ↪ artifacts/scan-util evaluate -i /input/
    ↪ sample.json -o /input/sample-ae.acc.json
```

Results: The aggregation result will become available as `sample-ae.acc.json` inside the `sample` subdirectory. The total number of clients, status and version distribution, offered key exchange algorithms, comment strings, and other evaluation criteria will match the data present within the `sample.json` file. Also, there is no difference between the `sample-ae.acc.json` and `sample.acc.json` files (aside from the evaluation start and end timestamps).

A.4.3 Troubleshooting

Address already in use. If an attack script is interrupted, some docker containers may not be cleaned up properly, blocking the server port permanently or for the duration of TIME-WAIT (1 min. on Linux, 30 sec. on MacOS).

Please follow these steps in this case:

1. Run the script `cleanup-system.sh`. This will stop and remove any pending Docker containers.
2. If the problem persists, wait for up to 4 minutes.

System Reset. To fully clean up the Docker containers and images, you can run `cleanup-system.sh --full`.

A.5 Notes on Reusability

The proof-of-concept code (`pocs/`) has been kept short for simplicity and is thus not modularized for reusability. However, the artifacts may serve as a template for other MitM attacks on network protocols like SSH.

The Docker files for the evaluated SSH implementations (`impl/`) may be generally useful in other research on SSH.

Improvements to Wireshark for better dissection of SSH protocols (not included in these artifacts) have been submitted and accepted upstream and will be available in a future version of Wireshark.

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/usenixsec2024/.