Artifact Evaluation Submission

Artifact Abstract

We are submitting two artifacts for review, both of which will be released publicly at our Github (https://github.com/breakerspace/). As we are still in the process of responsible disclosure, the Github repository is still private, but we are sharing the contents of the repository in full here for the purpose of the artifact evaluation.

The two artifacts are Python scripts used for the detection of abusable four-tuple residual censorship (`residual_censorship_scanner.py`) and for ethically triggering residual censorship from any arbitrary source IP address (`sp3_send.py`): what we used to test the efficacy of the attack.

Note that to experience our attack and exercise the full usage of both, you will need to have vantage points located inside (at least) one of the censored regimes we tested (Iran, Kazakhstan, and China) and (at least) one located outside of the censoring regimes. Obtaining vantage points inside these censored countries can be very difficult. To assist in the artifact evaluation, we are willing to coordinate configuring our vantage points to assist your testing. Otherwise, below we have noted how you can perform artifact evaluation without needing direct access to such a vantage point.

Disclaimer

Both of our artifacts will are designed to trigger censorship for large nation-states. For the purpose of artifact evaluation, there is no risk if you use them from uncensored vantage points to other uncensored vantage points, but please understand the risks of using them on vantage points within censored countries before doing so.

Artifact 1: Identifying Abusable Four-Tuple Residual Censorship

The `residual_censorship_scanner.py` script is used to identify abusable four-tuple residual censorship (null-routing censorship that can be used to perform this attack). The script is designed to be run from a client (located either inside or outside a censored regime) to an echo server located on the other side of the censor. The script attempts to trigger censorship a few different ways (notably without properly completing a 3-way handshake), and then sends various follow-up test packets and checks if the server responds to them. In total, it performs 6 tests; if all six result in censorship, the censorship system is likely abusable.

The script is designed specifically for an echo server on one side and expects a censor in the middle that performs four-tuple null-routing censorship.

To test this full functionality, you will need to obtain a vantage point within Iran or Kazakhstan. We have vantage points in Kazakhstan: if you are interested in coordinating, we are happy to re-open an echo server on our vantage point in Kazakhstan for you to test the functionality. We can coordinate through hotcrp or through the PC to maintain AE reviewer anonymity.

See the included README.md for the full usage instructions on the script.

Artifact Evaluation

To check this artifact, obtain a Linux vantage point located outside of a censoring regime (such as an EC2 machine) and follow the setup instructions in the README. Run the script and give it the IP address and port of an echo server located inside a censored regime (which we can provide), and confirm that the script identifies the four tuple residual censorship as abusable.

Alternatively, you can test the script between two machines that do not have a censor between them. Running the script between two EC2 machines gives the following expected output:

```
Summary:
- 3-way handshake, tested with a well-formed innocuous query on PSH+ACK packet, with a good seqno/ackno: no censorship.
- 3-way handshake, tested with a malformed innocuous query on PSH+ACK packet, with a good seqno/ackno: no censorship.
- No 3-way handshake, tested with a well-formed innocuous query on PSH+ACK packet, with a good seqno/ackno: no censorship.
- 3-way handshake, tested with a SYN packet, with a good seqno/ackno: no censorship.
- No 3-way handshake, tested with a SYN packet, with a good seqno/ackno: no censorship.
- No 3-way handshake, tested with a SYN packet, with a different seqno/ackno: no censorship.
No residual censorship detected.
```

When run to a vantage point located inside of Kazakhstan:

```
Summary:
- 3-way handshake, tested with a well-formed innocuous query on PSH+ACK packet, with a good seqno/ackno: censored.
- 3-way handshake, tested with a malformed innocuous query on PSH+ACK packet, with a good seqno/ackno: censored.
- No 3-way handshake, tested with a well-formed innocuous query on PSH+ACK packet, with a good seqno/ackno: censored.
- 3-way handshake, tested with a SYN packet, with a good seqno/ackno: censored.
- No 3-way handshake, tested with a SYN packet, with a good seqno/ackno: censored.
- No 3-way handshake, tested with a SYN packet, with a different seqno/ackno: censored.
Abusable residual censorship detected.
```

Artifact 2: Triggering Censorship with Spoofed Source IP Addresses

The second artifact we are submitting is the script we used to launch the attack against ourselves: ethically triggering censorship with spoofed source IP addresses. The script `sp3_send.py` relies on a public instance of the amazing (SP)^3 project (see https://github.com/willscott/sp3). SP3 is a service that allows a client to *consent* to receiving spoofed traffic. If you set up an SP3 server yourself on a vantage point without egress filtering, you can configure the script to use that; otherwise, it will default to using a public instance of SP3 located in the University of Washington.

To use this script to test the attack, you *must* use it from a machine located *inside* a censored regime. This is because this attack relies on the attacker and victim client being on the same side of the censor; as the public instance of SP3 is located in the United States, this means our victim client must be located outside censored regimes and victim server must be located inside censored regimes. As SP3 requires the recipient of spoofed traffic to consent to receiving traffic, this means the server inside the censored regime must run the script.

When you run the script, that machine will connect to SP3 with a websocket connection to consent to receiving spoofed traffic and then give SP3 'trigger packets' (a sequence of packets that elicit a censorship response) to send. We used this script to test the attack by setting the source IP address of the trigger packets to be vantage points we control in uncensored countries, triggering the attack by running `sp3_send.py` inside a censored country, and then attempting to make innocuous requests to the server from those vantage points. If the censor impedes this innocuous communication, we know the attack succeeded.

See the included README.md for the full usage instructions on the script.

Artifact Evaluation

To test the functionality of this artifact, obtain a Linux vantage point located outside of a censoring regime (such as an EC2 machine) and follow the setup instructions in the README. Run the `sp3_send.py` script on that machine with `tcpdump` running, and confirm you receive a `SYN` packet, followed by a `PSH+ACK` packing containing a payload (which will change depending on your chosen protocol). We note that SP3 does have some minor limitations to the source IP addresses it can send from; if you do not see traffic, try different source IP addresses. This packet sequence is already known to be a censorship trigger (see [11]), so seeing this traffic from a spoofed IP address is enough to validate this artifact.

If you want to go further and experience this attack in full yourself, you will need a vantage point located both outside and inside a censored regime. Obtain a vantage point inside Kazakhstan, Iran, or China (of the three, we have found KZ to be the easiest to safely find vantage points). Start an HTTP server on the vantage point inside the censored regime. Run the script inside the censored regime, and supply it the public IP address of your vantage point outside the censored regime. Issue innocuous follow-up requests from your client vantage point to the server you started, and confirm that the censor disrupts your requests.

Note that to prevent abuse (and limit the load on SP3), we developed the script to only trigger censorship for a single given source port. If testing four-tuple residual censorship, this will only block a single four tuple (source IP address, source port, destination IP address, destination port). To test if the resulting four-tuple residual censorship is affecting you, we can make requests that originate from the affected source port (such as with `curl` using the `--local-port` option, such as: `curl <ip-address-of-machine-in-censored-regime>:<port> --local-port 22222`.)

If obtaining vantage points located within the censored countries is prohibitive but you still want to try to see the attack live, we can coordinate running the script from our vantage point to attempt to block a vantage point you control from issuing innocuous requests to our vantage point.

Responsible Disclosure

As we noted in our paper, responsibly disclosing these findings is difficult, but we are in the process of communicating with several regional CERTs to try to disclose this attack.