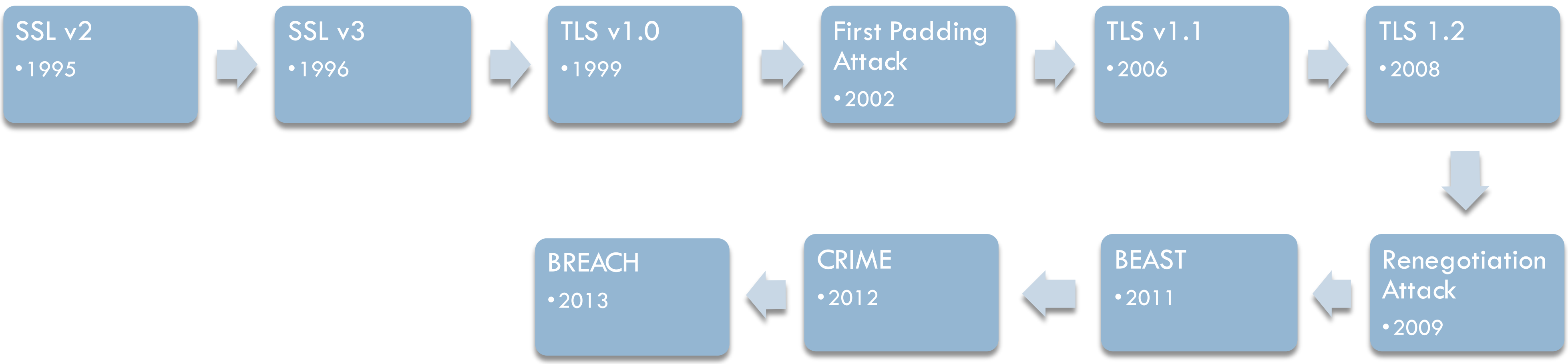
SSL/TLS: HISTORY AND VULNERABILITIES

Ernest Kushevski

History

- Father of SSL Dr. Taher ElGamel
- First public release as SSL v2 in 1995
- Many initial flaws
- SSL v3 released in 1996 with significant improvements
- TLS 1.0 released in 1999 with minor improvements
- TLS 1.1 released in 2006
- Most current version TLS 1.2 released in 2008

Timeline


SSL v2
• 1995
SSL v3
• 1996
TLS v1.0
• 1999
First Padding Attack
• 2002
TLS v1.1
• 2006
TLS 1.2
• 2008
Renegotiation Attack
• 2009
BEAST
• 2011
CRIME
• 2012
BREACH
• 2013

Overview of SSL/TLS Process

- SSL/TLS is an application/presentation layer protocol
- Consists of two layers
- Lower layer: record header and encrypted MAC and message
- Upper layer: Handshake, Change Cipher Spec, Alert, Application Data

Handshake Protocol

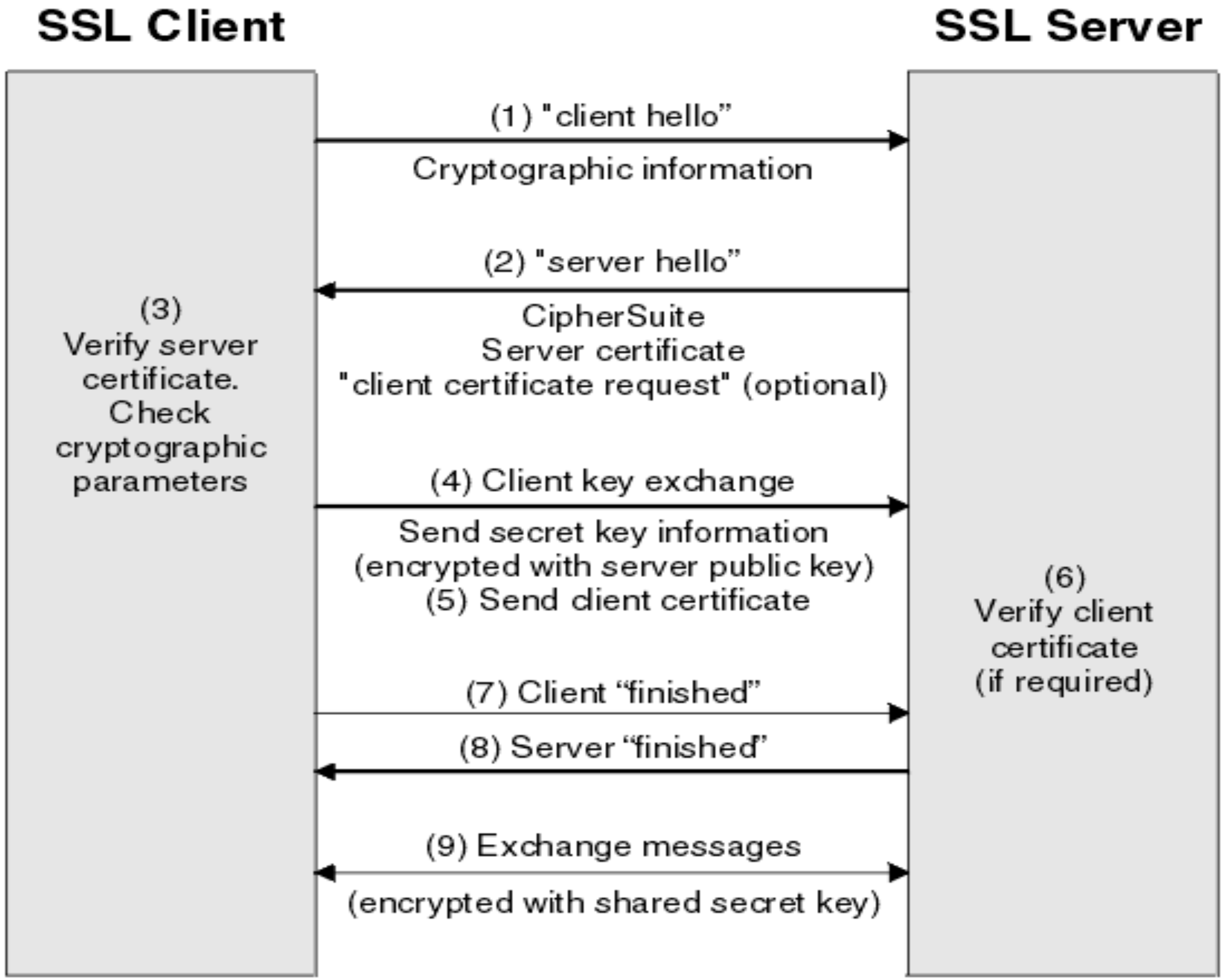
- Consists of 4 rounds
- Enables peers to agree on keys, ciphers, and MAC algorithms
- Begins when Client sends ClientHello with random number, a list of supported cipher suites, and compression methods
- Server responds with ServerHello, certificate, protocol version, chosen cipher suite, compression method, and random number

Handshake Continued

- Server sends ServerHelloDone to indicate the handshake is complete
- Client responds with a ClientKeyExchange message, and sends a pre-master secret encrypted with the server's public key
- The server and client use this and the previously sent random numbers to calculate the Master Secret which they now use to encrypt all messages

ChangeCipherSpec

- The client sends ChangeCipherSpec to confirm that all data is henceforth to be encrypted with a Finished message that contains the hash and the MAC of the previous handshakes
- The server responds with its own Finished message



http://sslnews.com.ua/book_ssl_tour-russian.html

Renegotiation Attack

- Discovered in 2009 by Marsh Ray and Steve Dispensa
- Vulnerability in TLS renegotiation feature
- Renegotiation used when attempting to authenticate while in SSL session
- Attacker initiates session with server
- Hijacks connection between client and server by holding handshake packets sent by client
- Requests authentication from server
- Presents the clients handshake instead
- Server now believes the attacker is the client
- Mitigation requires turning off feature or implementing a library extension

Renegotiation Continued

Step 1: Attacker positions himself in between the client and server prior to first TLS handshake

Step 2: Client begins TLS handshake with server, attacker holds these packets

Step 3: Attacker undergoes his own TLS handshake with server

Step 4: Attacker triggers renegotiation request with server

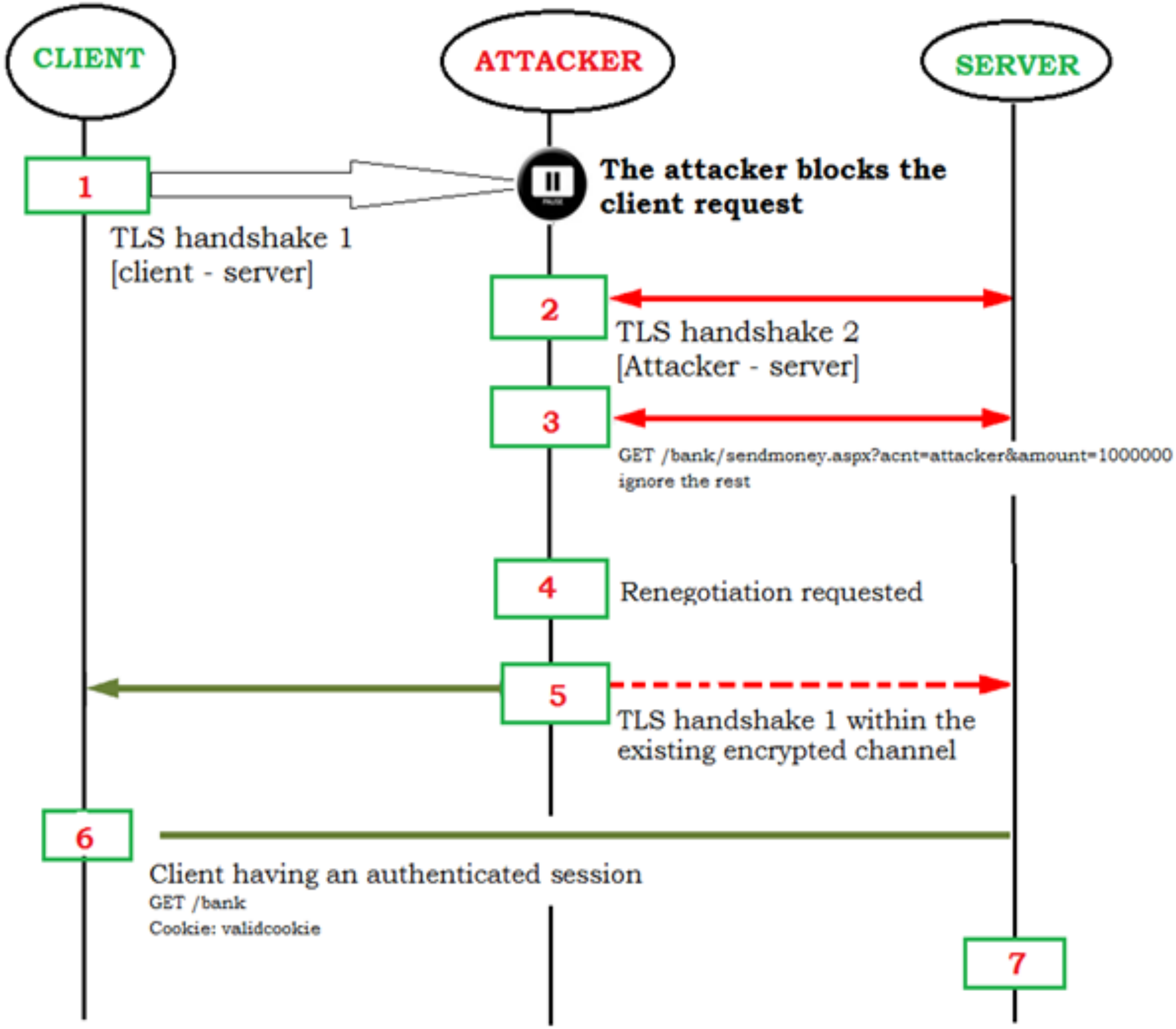
Step 5: During the renegotiation request attacker uses the client's initial handshake to communicate with the server

Step 6: The attacker who now acts as a proxy can mount an attack against the application layer protocol like HTTPS

Step 7: For instance he can append his is own GET request to whatever the client sends: e.g. Attacker sends GET /pizza?toppings=pepperoni;address=attackersaddress HTTP/1.1

X-Ignore-This **and client sends** GET /pizza?toppings=sausage;address=victimssaddress HTTP/1.1

Cookie: victimscookie , the two requests get glued together and a pizza is sent to the attacker



http://resources.infosecinstitute.com/ssl-attacks

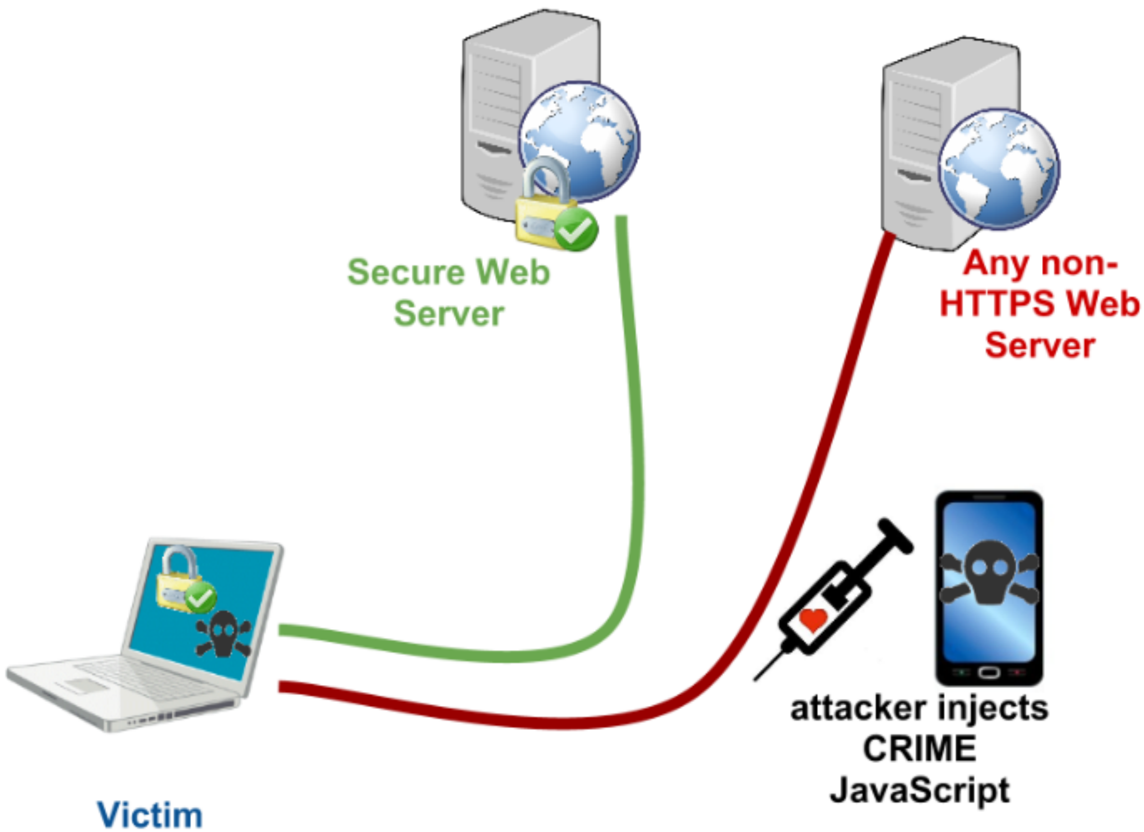
BEAST Attack

- BEAST (**Browser Exploit Against SSL/TLS**) is a vulnerability in the way cipher block chaining is implemented in pre-TLS 1.1 versions
- Thai Duong and Juliano Rizzo discovered in 2011
- The initialization vector (IV) used is the ciphertext of the previous block
- An attacker could use a chosen plaintext attack since IV is predictable
- The attacker will keep trying different plaintexts until he is able to decrypt

CRIME/BREACH

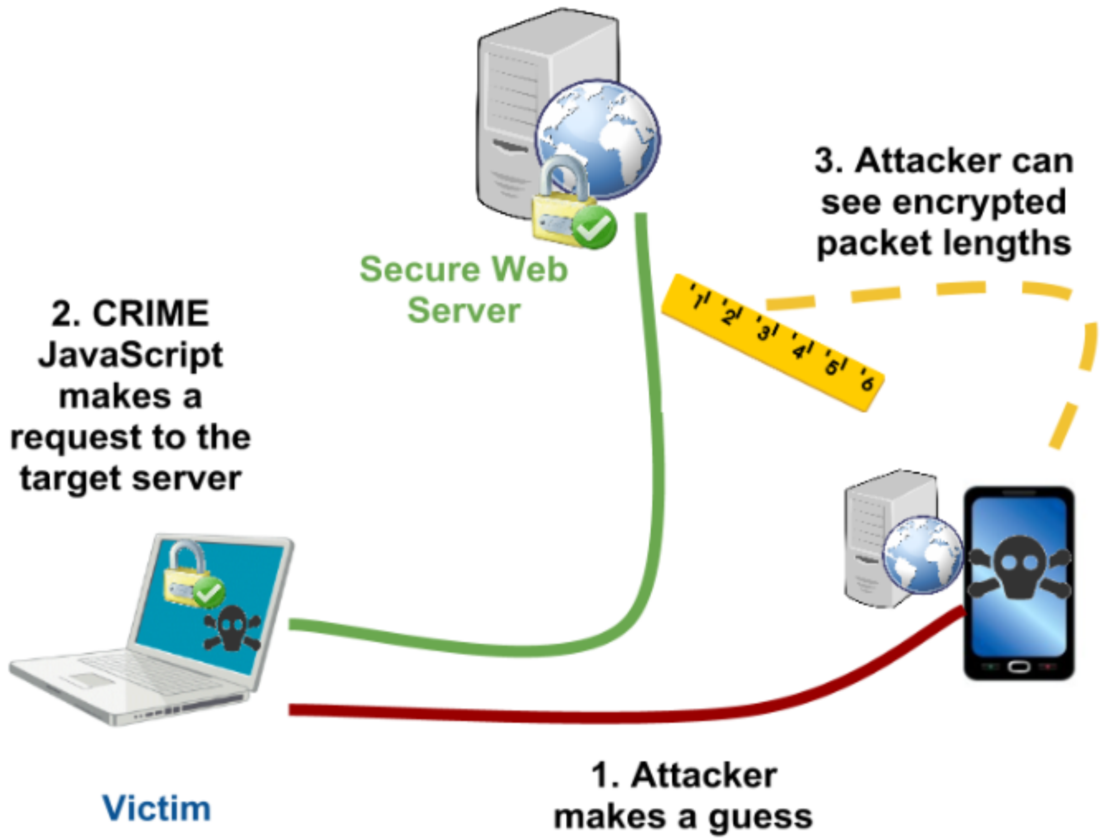
- Initially described in theory by John Kelsey in 2002
- CRIME (**Compression Ratio Info-leak Made Easy)** attack exploits compression used in TLS
- Compression works by reducing redundancies in data
- For instance, google and goggle
- In instances where compression is used it is possible to see changes in size
- A chosen plaintext attack could be mounted and file sizes observed to see if match is found
- CRIME could be mitigated by turning compression off
- BREACH (**Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext)** was the same compression-based attack against the HTTP protocol itself
- Difficult to mitigate because HTTP compression is widely used

CRIME injection



https://docs.google.com/presentation/d/11eBmGiHbYcHR9gL5nDyZChu_-lCa2GizeuOfaLU2HOU/edit?pli=1#slide=id.g1eb6c1b5_3_6

CRIME in a slide



https://docs.google.com/presentation/d/11eBmGiHbYcHR9gL5nDyZChu_-ICa2GizeuOfaLU2HOU/edit?pli=1#slide=id.g1eb6c1b5_3_6

Padding Attacks

- Original attack discovered by Serge Vaudenay in 2002
- Reliant the server to leak information on the success/failure of a padding check
- Attacker injects his own plaintext and confirms whether a padding or MAC error has occurred
- This allows the attacker to decrpyt/encypt messages without knowledge of the key
- TLS architects responded initially by changing the error message generated when an error occurs
- The timing differential remained

Padding Attacks Continued

- By noting the difference in time required to decrypt, attackers could obtain the same information as before
- TLS 1.2 requires that even if there is a padding error, the server assumes no padding and calculates MAC of whole message
- Very small timing differential still exists
- Lucky 13 Attack

Padding Attacks Continued

- Step 1: Attacker truncates the ciphertext message so that the plaintext to be decrypted is in the last block where padding is normally expected
- Step 2: Then the attacker modifies the second to last encrypted block, the last byte of which is XOR'ed with a guess of the plaintext
- Step 3: CBC decryption will attempt to decrypt that byte and if it is successful it will generate 0x00 which in turn generates a MAC error
- Step 4: A MAC check in TLS 1.0 takes longer than a a padding error to create an alert and thus the attacker can determine if his padding guess was successful
- Step 5: If the attacker has established that the last byte of the modified ciphertext produces valid padding with a particular value and he knows the plaintext value is 01 (this is the value of the padding) then he can reverse the operation and confirm the intermediate state
- Step 6: Using the intermediate state he can submit the actual ciphertext and decrypt that into plaintext

Conclusion/Mitigations

- BEAST is has been successfully mitigated for versions 1.1+ and is also difficult to execute
- CRIME has been mitigated in new browser versions
- BREACH is still possible and difficult to mitigate
- Renegotiation attacks can be mitigated by using a library that includes an extension proposed in RFC 5746
- Padding attacks are still possible

Questions?