Drown Attack

Stephen O’Kennedy
15305690

Acknowledgements to Aviram et al.[3]

What is DROWN

- Decrypting RSA using Obsolete and Weakened eNcryption
- Cross Protocol Attack
- Allows for the decryption of intercepted TLS connections
- Can be used to perform
  - A man in the middle attack
  - Eavesdropping
- Only focuses on a single session
- Affects 33% of all servers, prior to publication of report

How it works

- Uses a variant of the Bleichenbacher padding Oracle attack

- Creates specially crafted connections to an SSLv2 server using the same private key

- Exploits SSLv2 flaws and OpenSSL bugs

Bleichenbacher's attack

- Exploits the fact that RSA ciphertexts should decrypt to plaintexts compliant with the PKCS#1 v1.5 padding format
- If an RSA decrypts to an invalid PKCS#1 v1.5 message, it might naturally leak this via
  - Error message
  - Closing the connection
  - Taking longer to process the error condition
- Leaked information about the plaintext that can be modelled as a cryptographic *oracle* for the decryption process

OpenSSL Bug

- SSLv2 Has been removed from OpenSSL v1.1.0+
- OpenSSL servers do not respect the cipher suites advertised in the ServerHello message
- Client can select an *arbitrary* cipher suite in the ClientMasterKey message and force the use of ex- port cipher suites even if they are explicitly disabled in the server configuration
- The SSLv2 protocol itself was still enabled by default in the OpenSSL standalone server

DROWN Variations

- 2 Types of DRAWN Attack
  - General DROWN Attack
    - Passive attack
  - Special DROWN Attack
    - Man in the Middle Attack

Breaking TLS with SSLv2

- How an SSLv2 Oracle can break TLS connection
- Attack Scenario
  - Target Server accepts TLS connections
  - The same RSA public key as the TLS connections is also used for SSLv2

Breaking TLS with SSLv2 Contd. Attacker's Position

- Able to passively eavesdrop on traffic between the client and server

- Records RSA-based TLS traffic

- Can't perform Man in the middle attacks

- Can expect to decrypt one out of 1,000 intercepted TLS connections

    - A decrypted TLS connection might contain a cookie or plaintext password

Breaking TLS with SSLv2 Contd. Attacker's Position

- To collect 1,000 TLS connections
  - Target: Server
    - DNS spoofing
    - BGP hijacking attack to redirect traffic transparently through themselves
  - Target: Client
    - Embed or inject malicious JavaScript on an site and force repeated connections
  - Needs to force an error with TLS connection to get a fresh handshake

Breaking TLS with SSLv2 Contd.

- Generic SSLv2 Oracle
    - The attacks make use of a padding oracle that can be queried on a ciphertext and leaks information about decrypted plaintext
        - Can abstractly models the information gained from an SSLv2 server's behaviour
- Oracles reveal many bytes of plaintext

DROWN Attack Template

- The attacker will use an SSLv2 oracle $\mathcal{O}$ to decrypt a TLS ClientKeyExchange.
- 2 problems with the behaviour of $\mathcal{O}$

    1. A TLS ciphertext transmitted in a TLS key exchange decrypts to a 48-byte premaster secret. No SSLv2 has 48 bit key strength

    2. $\mathcal{O}$ is very restrictive, since it strictly checks the length of the unpadded message. Would require 12 million oracle queries

DROWN Attack Template

Attack Flow

- Overcomes the $\mathcal{O}$'s problems by:

    1. Collecting many encrypted TLS RSA key exchange messages

    2. Tries to convert the intercepted TLS ciphertexts containing a 48-byte premaster secret to valid RSA PKCS#1 v1.5 encoded ciphertexts containing messages of length appropriate to the SSLv2 oracle $\mathcal{O}$

        - Accomplish this by taking advantage of RSA ciphertext malleability and a technique of Bardou et al [1]

    3. Once a valid SSLv2 RSA ciphertext is obtained, we can continue with a modified version of Bleichenbacher's attack, and decrypt the message after many more oracle queries

    4. Can then transform the decrypted plain- text back into the original plaintext, which is one of the collected TLS handshakes

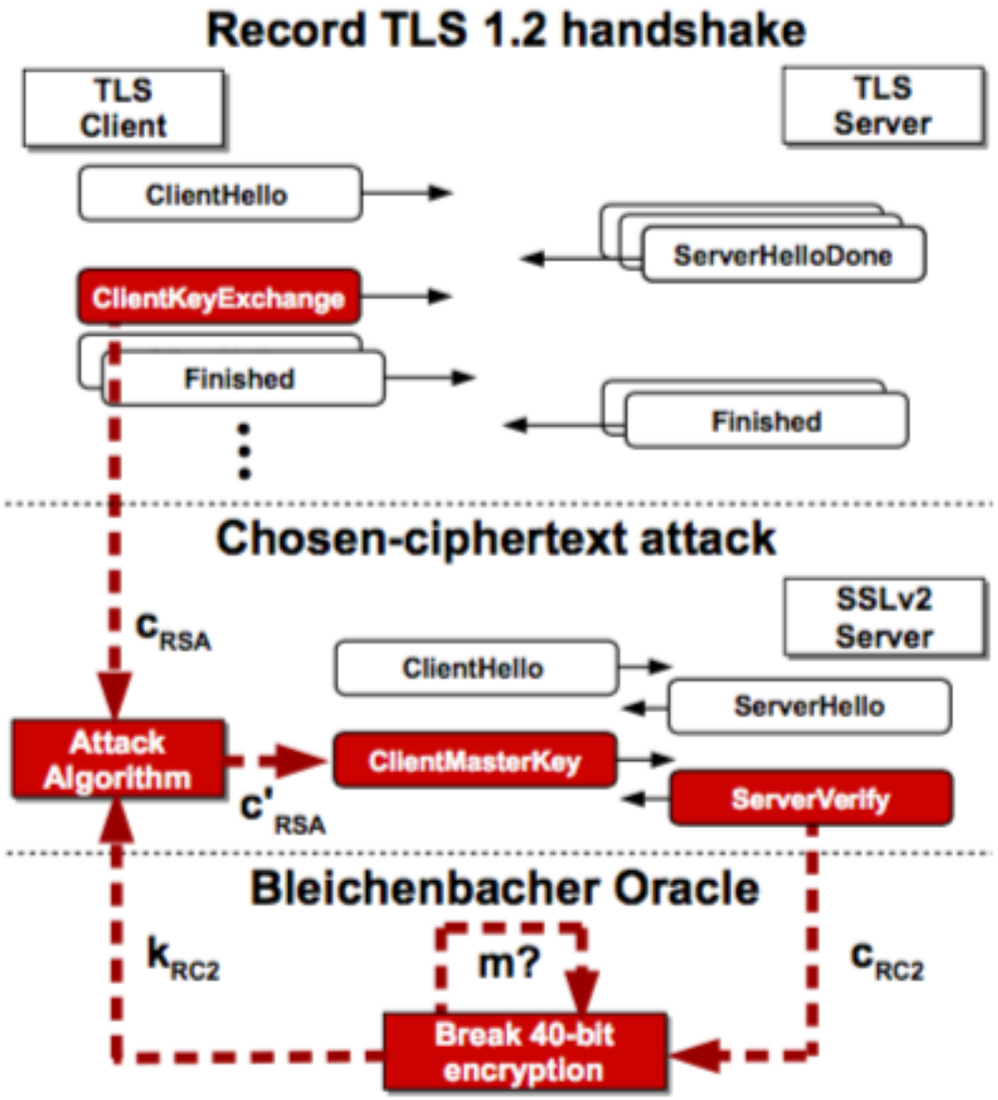
General DROWN
CVE-2016-0800

- Any correct SSLv2 implementation that accepts export grade cipher suites can be used as an oracle
- SSLv2 is vulnerable to a direct message side channel vulnerability exposing a Bleichenbacher oracle to the attacker
- The vulnerabilities follows 3 properties of SSLv2

1. The server immediately responds with a ServerVerify message after receiving the ClientMasterKey message
    - Includes the RSA ciphertext, without waiting for the ClientFinished message
    - Proves that the client knows the RSA plaintext
2. When choosing 40-bit export RC2 or RC4 as the symmetric cipher, only 5 bytes of the master_key are sent encrypted using RSA, and the remaining 11 bytes are sent in cleartext
3. Server implementation that correctly implements the anti-Bleichenbacher counter- measure and receives an RSA key exchange message with invalid padding will generate a random premaster secret and carry out the rest of the TLS handshake using this randomly generated key material

General DROWN
Validating RSA ciphertexts

- The attacker sends a ClientMasterKey message
- The server responds with a ServerVerify message, which contains the challenge encrypted using the server_write_key
- The attacker performs an *exhaustive search* over the possible values of the 5 bytes of the master_key
- Then computes the corresponding server_write_key and checks whether the ServerVerify message decrypts to the challenge
- The attacker re-connects to the server with the same RSA cipher text
- The server responds with another ServerVerify message that contains the current challenge encrypted using the current server_write_key
- If the decrypted RSA cipher- text was valid, the attacker can directly decrypt a correct challenge value from the ServerVerify message by using the master_key
- Otherwise, if the ServerVerify message does not correctly decrypt to the challenge, the RSA ciphertext was invalid
    - The attacker knows the mastery was generated at random



General DROWN
TLS Decryption Attack
Attack Scenario

- A server that accepts TLS connections from clients using an RSA public key that is exposed via SSLv2
- The server supports export cipher suites for SSLv2
- The server implements the recommended countermeasure against Bleichenbacher's attack in all protocol versions, including SSLv2
- The attacker needs access to computing power sufficient to perform a 2^50 time attack, mostly brute forcing symmetric key encryption

General DROWN
TLS Decryption Attack
Constructing the Attack

1. Attacker passively collects 1,000 TLS handshakes from connections using RSA key exchange

2. The attacker then attempts to convert the intercepted TLS ciphertexts containing a 48-byte premaster secret to valid RSA PKCS#1 v1.5

    - Sends the modified ciphertexts to the server using fresh SSLv2 connections with weak symmetric ciphers and uses the ServerVerify messages to deduce ciphertext validity

    - For each RSA ciphertext the attacker needs to perform a brute force attack on the weak symmetric cipher

    - Can to obtain a valid SSLv2 cipher text after ~10,000 oracle queries or 20,000 connections to the server

3. Once the attacker has obtained a valid SSLv2 RSA ciphertext $m1$

    - A shifting technique is used find an integer $S1$ such that $m2=m1 \cdot 2^{-40} \cdot S1$

4. The attacker then applies the shifting technique again to find another integer $S2$ such that $m3=m2 \cdot 2^{-40} \cdot S2$

5. The attacker can continue with use an adapted Bleichenbacher iteration technique

6. Can then transform the decrypted plain- text back into the original plaintext

General DROWN
TLS Decryption Attack
Cost of the Attack

- Most expensive computationally part of the attack is breaking the 40-bit attack

- GPU implementation performed around 515MH/s, where MH measures the calculation of an MD5 hash and the RC2 decryption

- The attack takes just under 8 hours, including startup and shutdown, using Amazon EC2 service for the cost of $440

Special DROWN Attack
CVE-2016-0703

- There was vulnerability in OpenSSLv2 handshake code that allowed for that creates a powerful Bleichenbacher oracle, and drastically reduces the amount of computation required to implement the attack

- Can cut the number of connections required by 50%

- Reduces the computational work to a negotiable amount

Special DROWN Attack
“Extras Clear” Oracle

- OpenSSL servers allowed the ClientMasterKey message to contain a clear_key_data bytes for *non-export* ciphers
- When present, the server substitutes them for bytes from the encrypted key
- Example:
  - Client chooses a 128-bit cipher and sends 16 byte key k[1].k[2],… k[16]
  - Contrary to protocol, it includes 4 null bytes of clear_key_data
  - OpenSSL generates this master_key [00 00 00 00 k[1] k[2] k[3] k[4] ... k[9] k[10] k[11] k[12]]
  - Enables a straightforward key-recovery attack against such versions
  - An attacker that has intercepted an SSLv2 connection takes the RSA ciphertext of the encrypted key and replays it in non-export handshakes to the server with varying lengths of clear_key_data

Special DROWN Attack
“Extras Clear” Oracle Contd.

- Example:
  - For a 16-byte encrypted key, the attacker starts with 15 bytes of clear key, causing the server to use the master_key [00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *k*[1]]
  - The attacker can brute force the first byte of the encrypted key by finding the matching ServerVerify message among 256 possibilities
  - Knowing the first byte, the attacker makes another connection with the same RSA ciphertext but 14 bytes of clear key, resulting in the master_key [00 00 00 00 00 00 00 00 00 00 00 00 00 00 *k*[1] *k*[2]]
  - The attacker already knows *k*[1], The can easily brute force the second byte
  - With only 15 probe connections and an expected $15 \cdot 128 = 1,920$ trial encryptions, the attacker learns the entire master_key for the recorded session
  - This session key-recovery attack can be directly converted to a Bleichenbacher oracle.

Special DROWN Attack
Attack Scenario

- Same as general attack
- The same RSA key pair used for TLS is also used on a server that is running a vulnerable version of OpenSSL

Special DROWN Attack
Constructing the Attack

1. Intercepts several hundred TLS hand- shakes using RSA key exchange

2. The attacker uses the fractional trimmers to convert the TLS cipher- texts into an SSLv2 conformant cipher text

3. Once a valid SSLv2 cipheertext is obtained, the attacker uses at the shifting technique to rotate the message by 25 bytes each iteration, learning 27 bytes with each shift. Takes several iterations to learn the entire plaintext

4. Then transforms the decrypted SSLv2 plaintext into the decrypted TLS plaintext

Special DROWN Attack
Attack Cost

- Using 40 fractional trimmers is a more efficient oracle attack

- Allows the attacker to recover 1 in 260 TLS keys using about 17,000 connections to the server

- Computation cost is so low that a full attack could be done on a single workstation

- Using the optimised version of this attack allows the Attacker to recover 1 in 100 at the cost of 27,000 connections

Man In the Middle

- Special Drown is fast enough to decrypt a TLS premaster secret online, during a connection handshake that a MITM(Man in the Middle) is possible

- A MITM can be used to compromise connections between modern browsers and TLS servers

    - Even those configured to prefer non-RSA cipher suites

Man In the Middle
Attack Scenario

- The MITM attacker impersonates the server
    - Sends a ServerHello message that selects a cipher suite with RSA as the key-exchange method
- The attacker uses special DROWN to decrypt the premaster secret
- The difficulty is completing the decryption and producing a valid ServerFinished message before user is timed out
    - Most browsers allow a handshake to last for 1 minute before timing out
- Using the fully optimised Special DROWN Attack is fast enough to not be timed out.

Vulnerable Websites

TCD.ie

Special DROWN Attack

Vulnerable Domains:	Vulnerable Because:
elib.tcd.ie *.elib.tcd.ie view certificate	134.226.14.55:443 vulnerable to CVE-2016-0703
lists.tcd.ie view certificate	134.226.14.105:443 vulnerable to CVE-2016-0703
mobiledevice.tcd.ie mobilereg.tcd.ie view certificate	134.226.14.85:443 vulnerable to CVE-2016-0703

These domains are vulnerable to
Man in the middle attacks

General DROWN Attack

Vulnerable Certificates:	Vulnerable Because:
autodiscover.tcd.ie email.tcd.ie go.tcd.ie mail.tcd.ie mail2.tcd.ie view certificate	134.226.254.66:443 supports SSLv2 export ciphers 134.226.254.67:443 supports SSLv2 export ciphers 134.226.254.68:443 supports SSLv2 export ciphers
password.tcd.ie view certificate	134.226.14.23:443 supports SSLv2 export ciphers
*.tchpc.tcd.ie view certificate	134.226.112.28:443 supports SSLv2 export ciphers
*.dental.tcd.ie view certificate	193.1.68.20:443 supports SSLv2
opendata.tcd.ie opendata2.tcd.ie opendatatest.tcd.ie popendata.tcd.ie view certificate	134.226.14.155:443 supports SSLv2

References

(1) BARDOU, R., FOCARDI, R., KAWAMOTO, Y., SIMIONATO, L., STEEL, G., AND TSAY, J.-K. Efficient padding oracle attacks on cryptographic hardware. In Advances in Cryptology–CRYPTO 2012. Springer, 2012, pp. 608–625.

(2) DROWN Attack. (2016). [online] Drownattack.com. Available at: https://drownattack.com/ [Accessed 4 Apr. 2016].

(3) Nimrod Aviram, Sebastian Schinzel, Juraj Somorovsky, Nadia Heninger, Maik Dankel, Jens Steube, Luke Valenta, David Adrian, J. Alex Halderman, Viktor Dukhovni, Emilia Käsper, Shaanan Cohney, Susanne Engels, Christof Paar and Yuval Shavitt. DROWN: Breaking TLS using SSLv2. Available at https://drownattack.com/. March 1, 2016.