Drown Attack

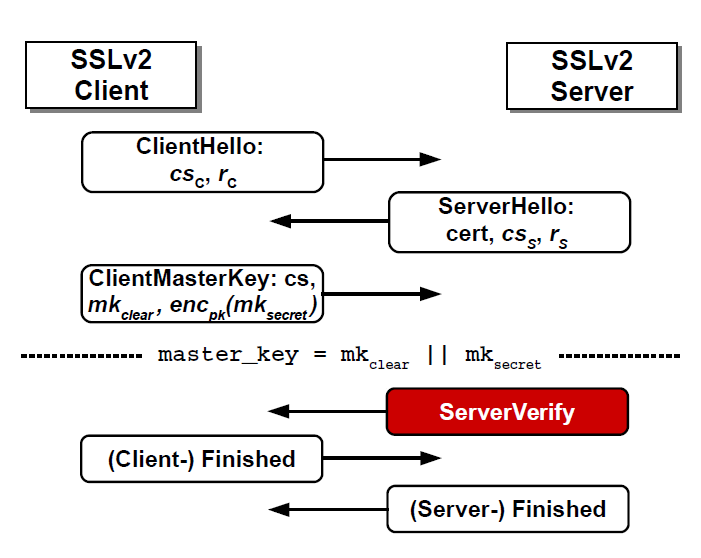
DROWN (Decrypting RSA with Obsolete and Weakened eNcryption) is a security attack that exploits vulnerabilities in SSLv2 to decrypt RSA encrypted connections. It targets servers that still support the obsolete and insecure SSLv2 protocol, even if they primarily use newer TLS versions. Many servers have SSLv2 enabled for backward compatibility.

Overview:

These days most modern servers don't support SSLv2 and use other protocols like TLS. Yet in 2016, out of 36 million HTTPS servers, 6 million support SSLv2 [1]. The attacker takes advantage of the SSLv2 handshake in order to decrypt the key ciphertext.

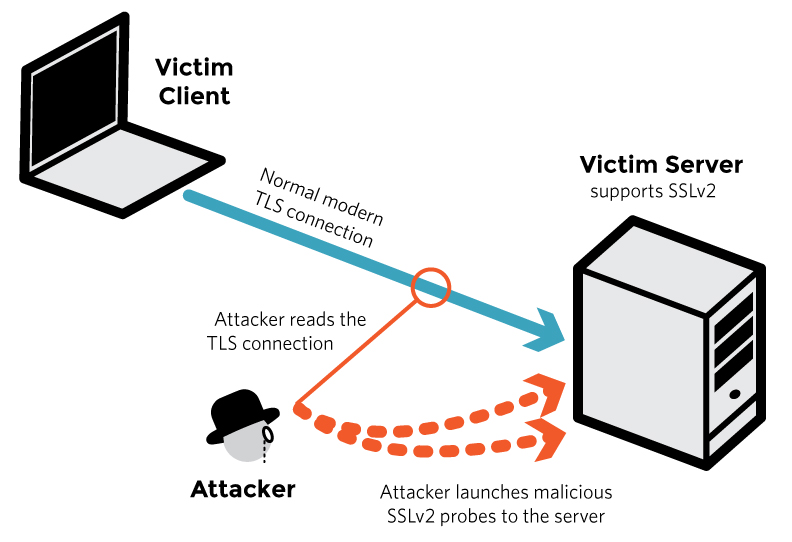
SSLv2 handshake:

In SSLv2 handshake, the client sends the clientHello message and receives the ServerHello message. The client then sends the master key encrypted with the server's key. At this point, the server encrypts the ciphertext and if the format of the message is valid, encrypts a new message with the symmetric key and sends it to the client. In case of not valid, It generates a random key and encrypts the message with that. This allows the attacker to figure out the validation of the format by sending a message twice. If the keys for the responses are different, it means the server, generated two different random keys, so the format was not valid.



In the Drown attack, the attacker uses SSLv2 protocol in order to break the TLS connection.

As shown in Figure below, the client that doesn't support SSLv2 sends a request to a server using TLS protocol. Although this connection is through TLS, the server might support SSLv2 too. Or in some cases, there might be another server supporting SSLv2 that shares the same key with the former server. In this case, the attacker intercepts and captures the traffic between the client and the server. Then it sends the traffic to the SSLv2 server multiple times, each time with a little modification in order to collect some information. Using this information it can then decrypt the RSA ciphertext. The decryption step is done using the Bleichenbacher attack.

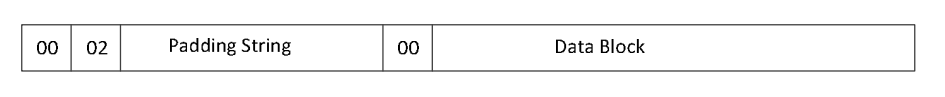


Details of decryption using Bleichenbacher:

bleichenbacher's padding oracle attack is an adaptive chosen ciphertext attack against PKCS#1 v1.5,the RSA padding standard used in SSL and TLS. It enables the decryption of RSA ciphertexts if a server distinguishes between correctly and

incorrectly padded RSA plaintexts, and was termed the “million-message attack”.

As shown in Figure below, in PKCS#1 v1.5 encryption padding data being encrypted has a specific format. When data is being sent to the server, it decrypts the data and validates the format. If the format has a problem the server returns 1, in case of proper format it returns 0.



In other words, the valid format would be:

2B <= m <= 3B-1 where B =

The attacker starts with ciphertext c0 and sends it to the oracle server. Based on the result it modifies the ciphertext step by step so that the possible solutions set become smaller. The new ciphertext would be:

c = (c0.) mod N = mod N

By receiving 1, the attacker modifies the ciphertext again. Otherwise, he can deduce for some value r, 2B <= (m0.)mod N < 3B. And can figure out the possible range form as below:

<= m0 <

Despite Bleichenbacher, which attacks TLS directly with a key length of 384 bits, Drown attacks deals with short secret for export-grade crypto which means the key length is 40 bits. Also in TLS, the server chooses the type of cipher suite, so we don't have information about the exact length of the key.

Implementation:

* Interception and record: Intercept network traffic and capture about 1000 TLS connections.
* Morph TLS connection: The connection captured is not compatible with SSLv2 oracle and it can not decrypt it. So it is needed to find the SSLv2 format for the ciphertext.
* Decrypt the key using bleichenbacher: send modified ciphertext multiple times in order to narrow down the possible answers and continue till only one solution remains.

Mitigation:

There are ways to mitigate the risk of this attack:

* Disable SSLv2 entirely: This prevents exploitation of SSLv2 vulnerabilities. Disable SSLv2 on both clients and servers where possible.
* Check public key reuse: Use different public/private key pairs for SSLv2 and TLS if both are required. Special DROWN relies on reusing keys.
* Patch OpenSSL: OpenSSL prior to 1.0.2f and 1.0.1r are vulnerable. Upgrade to newer patched OpenSSL versions.

References:

[1] Aviram, N., Schinzel, S., Somorovsky, J., Heninger, N., Dankel, M., Steube, J., Valenta, L., Adrian, D., Halderman, J.A., Dukhovni, V. and Käsper, E., 2016. {DROWN}: Breaking {TLS} Using {SSLv2}. In *25th USENIX Security Symposium (USENIX Security 16)* (pp. 689-