Team: Password Is Password

Mohit Jangid, Eric Lewantowicz, Joseph Shaffer

CSE 5473

4/22/2018

**Bleichenbarcher and POODLE SSLv3 Padding Attacks**

**POODLE Attack**

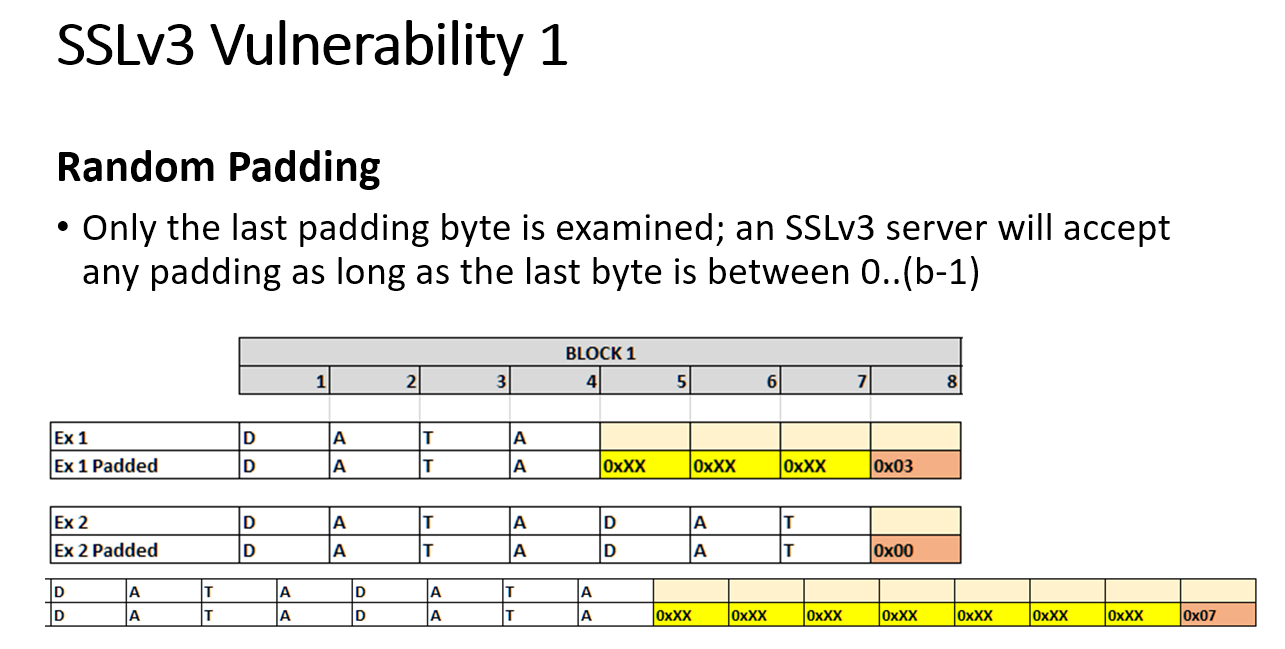
**Goals**

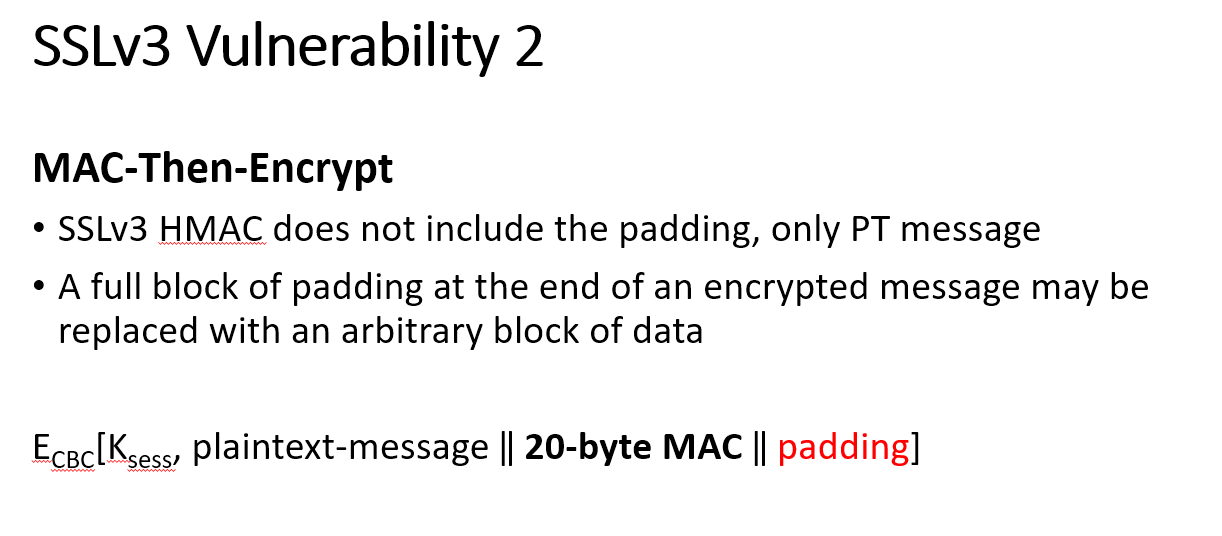
1. Learn the vulnerabilities of SSLv3 encryption scheme and the details of the POODLE attack.
2. Implement the SSLv3 protocol at a low-level from scratch in Python 2.7 with the Pycrypto module, with combined Client/Attacker communicating with Server Oracle over TCP stream sockets.
3. Write a GUI from scratch using Curses shell graphics module in order to provide an effective visualization of the attack simulation.
4. Consider possible defensive approaches and implement a defense against a POODLE attack.

The Padding Oracle On Downgraded Legacy Encryption attack was discovered by a Google Security team in 2014. It is a man-in-the-middle attack on the (at the time) deprecated but still supported SSLv3 encryption scheme using block CBC mode. Because many servers still supported SSLv3 in order to support legacy applications and browsers still employing it, an attacker could execute a “downgrade dance” to cause a browser and server to fallback to an SSLv3 encryption scheme. Using elements from the BEAST attack (see references), an attacker can inject an HTTPS request generator into a victim client’s browser using JavaScript injection from a complicit HTTP website. The attacker could then use the request generator to fashion plaintext messages, that combined with a secret cookie, would be encrypted with SSLv3 and then sent to an HTTPS server. The attacker intercepts the encrypted traffic, moves the ciphertext block containing the secret cookie to the end of the message, replacing a full block of padding, and then forwards the modified encrypted message to the server. The server acts as an oracle and disconnects the session if the padding or MAC is incorrect, and accepts the message if the padding is correct. This process allows the attacker to decrypt the contents of the message, i.e. the cookie, byte by byte.

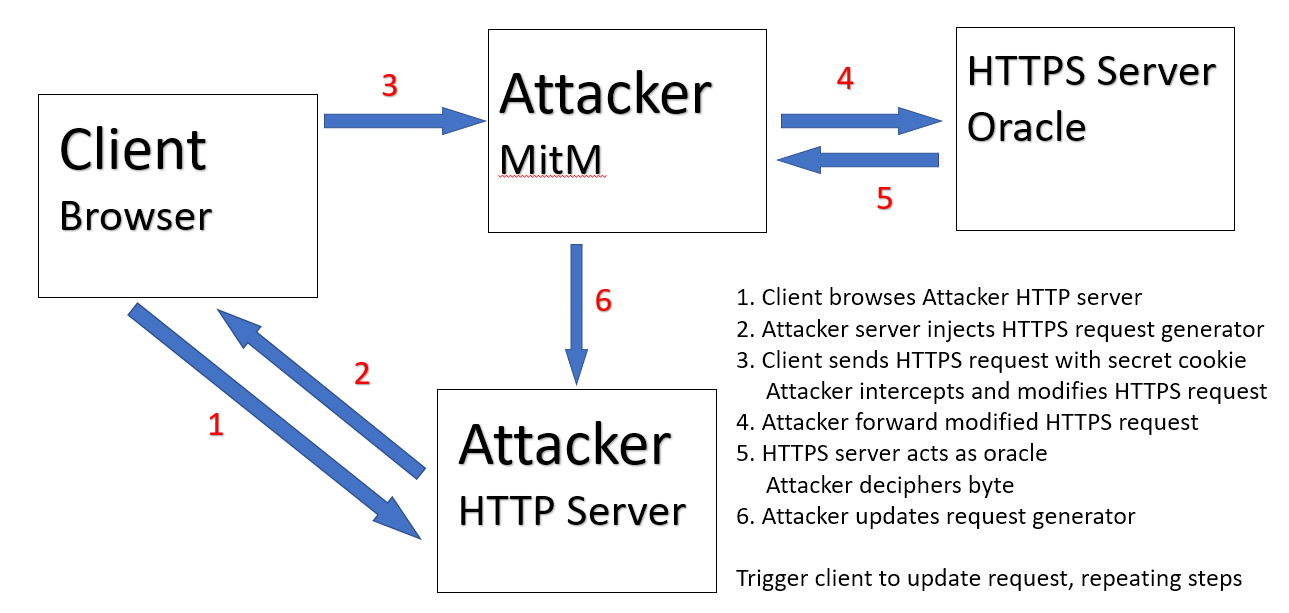
After discovery of the POODLE attack, the long-term solution was to eliminate SSLv3 support altogether in all browsers and servers, since the RC4 stream variation of SSLv3 has its own vulnerabilities and is not considered secure. However, several short-term mitigations were proposed and implemented in the short term, including anti-POODLE record splitting by the Opera browser. While we didn’t find any details on their record-splitting approach, the idea served as the seed for our own record splitting defense proposal.

**Key SSLv3 vulnerabilities that the POODLE attack exploits** (graphics from our PowerPoint presentation)





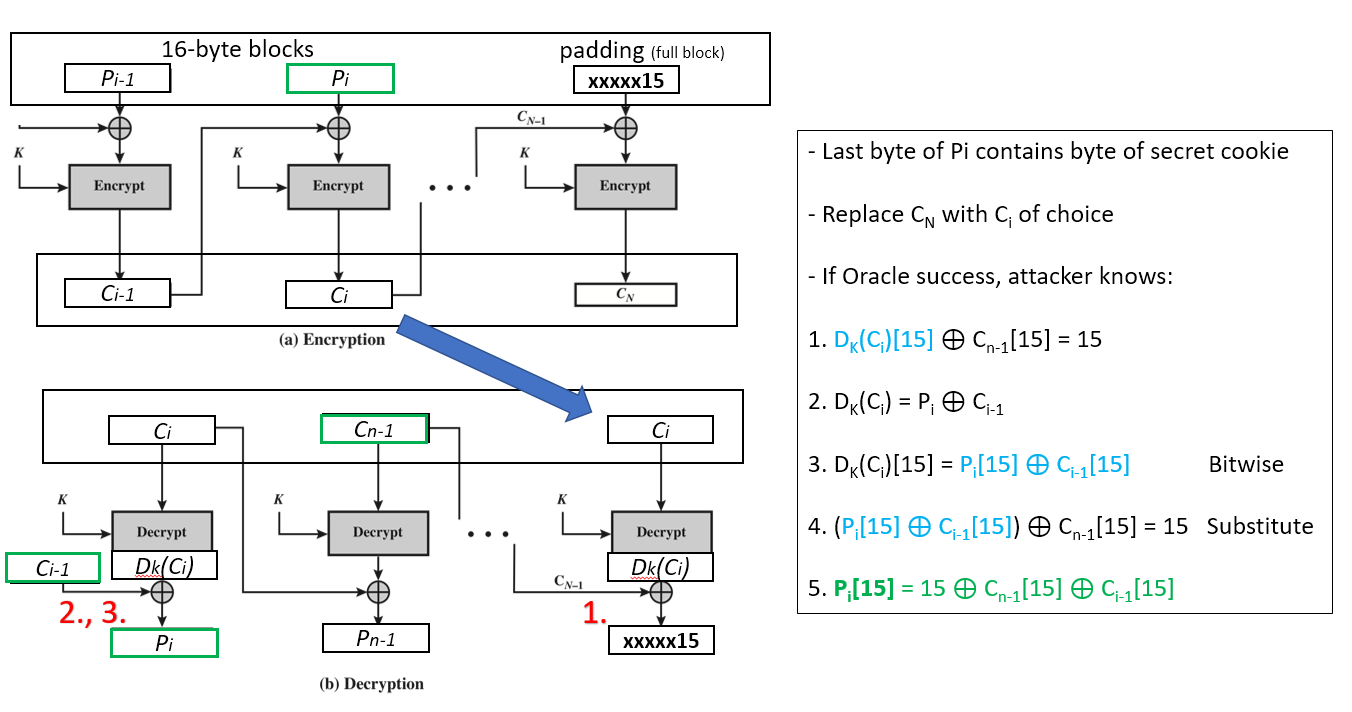
**Attack overview** (diagram from our PowerPoint presentation)



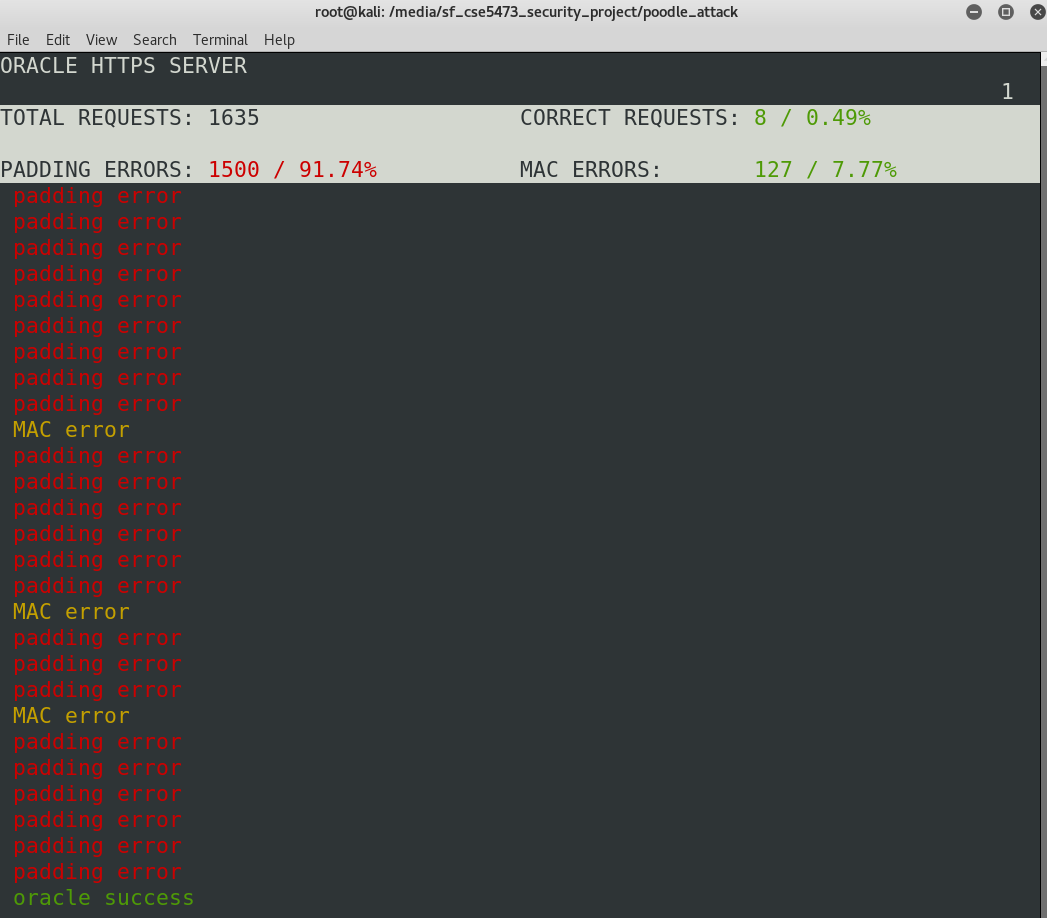
**Key Attack Steps**

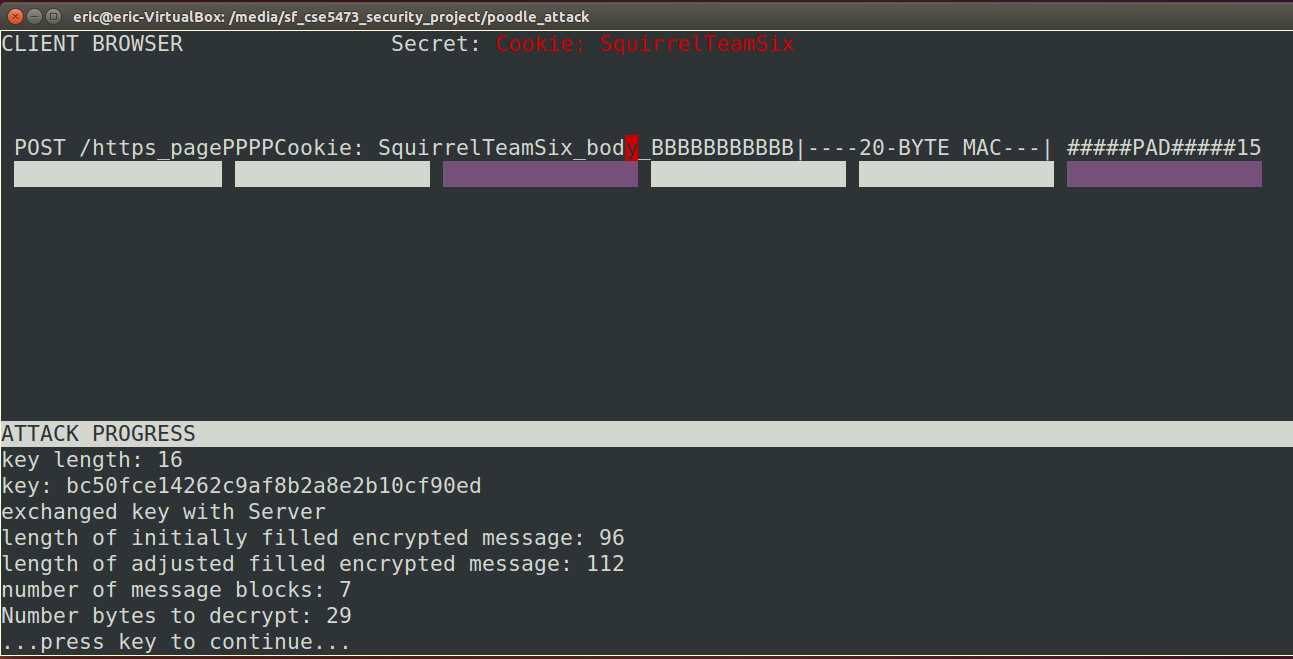
1. Add initial fill bytes to the HTTPS request path and body.
2. Intercept modified encrypted request, C0||C1||...Cn, and continue to add fill bytes to the HTTPS request body until the modified encrypted message increases by a block length. The IV is appended to the encrypted message as C0 from the client.
3. Key concept: now the attacker knows a full block of padding exists in block Cn, with no MAC contamination. This block can be replaced by a block Ci without affecting the MAC.
4. Replace block Cn with block Ci and forward the request to the server oracle.
5. If the server accepts the block with correct padding/correct MAC, the last byte of block Ci can be decrypted (see details in graphic below)
6. If the server rejects the block with incorrect padding/MAC, the attacker triggers the client to re-encrypt the same request. A new IV is used, so the encrypted contents are changed, Cn is again replaced with Ci, and the message is again sent to the server. This process is repeated until the server accepts the block.
7. After decrypting the last byte of Ci, attacker request generator removes one fill byte from the request body, and inserts an additional fill byte to the path. This shifts the cookie/data blocks one byte to the right while maintaining a full block of padding.
8. Repeat steps 4-7 until all of the cookie bytes are decrypted.

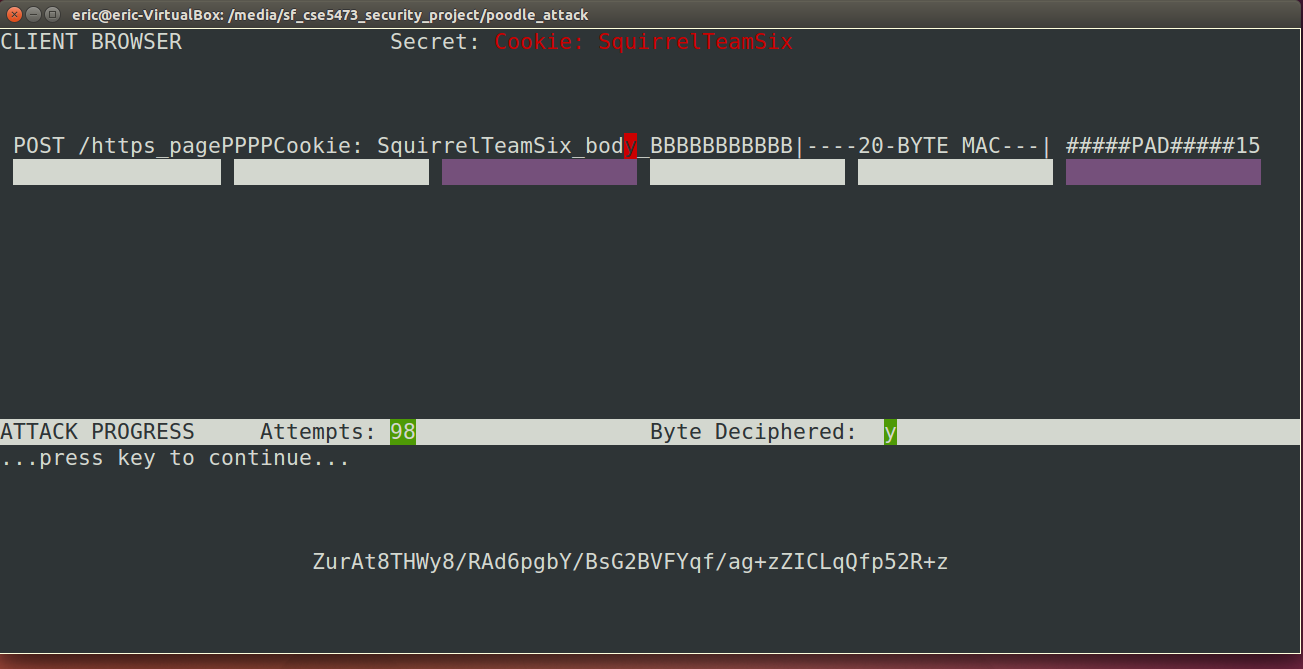
**Low-level attack process visualized** ((graphic from our PowerPoint presentation

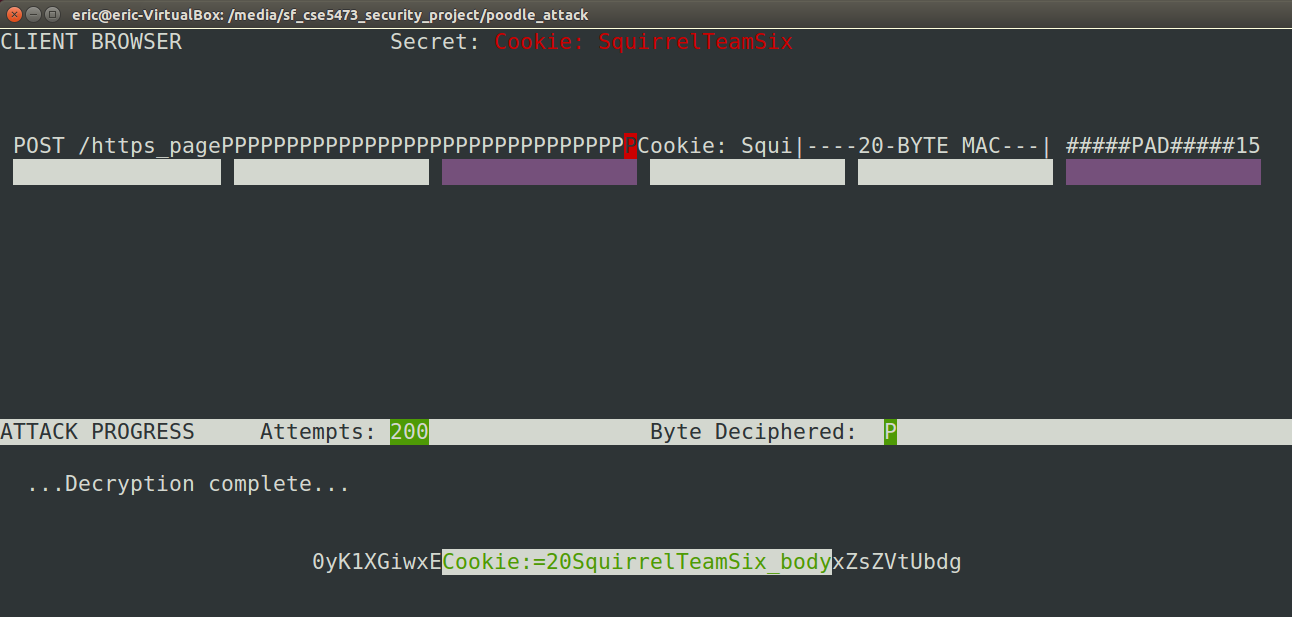


**Our Implementation and Visualization of the Attack and Defense using Curses shell graphics module**







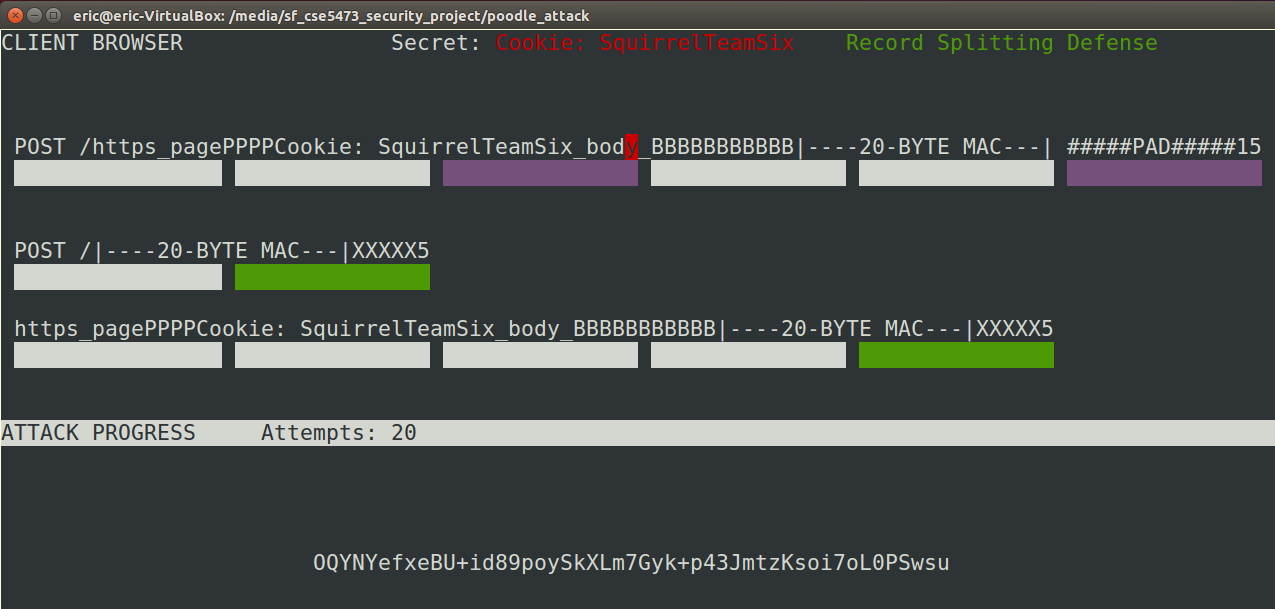


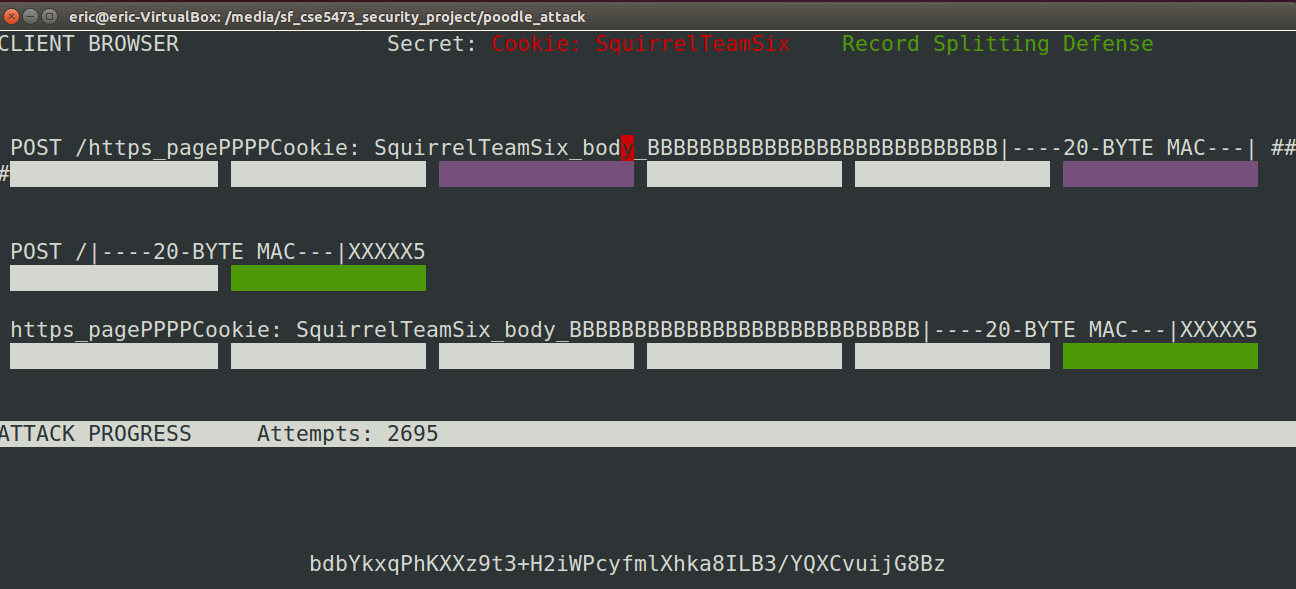
**Defense Proposal & Experiment**

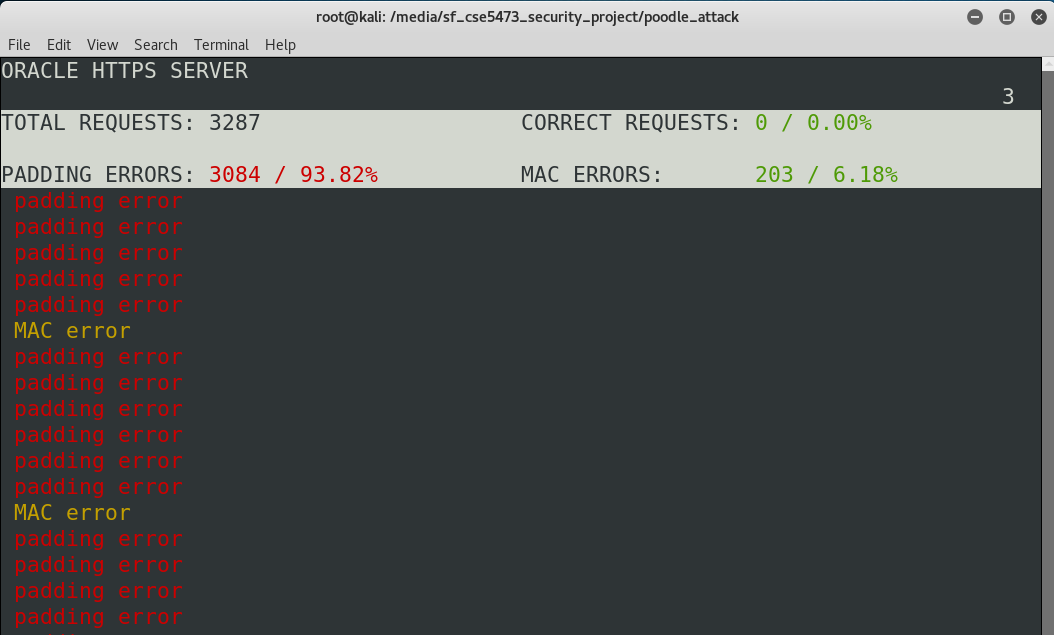
The attack is a very low-effort attack, with a successful response from the server occurring every 1/256 attempts, until an IV / Key combination produce a final padding byte in Ci that is decrypted to a value of 15 (x0F). The Opera browser implemented some temporary anti-POODLE record splitting whose details we didn’t find, but gave us the idea to split vulnerable records before encryption, in order to ensure that a full block of padding would never exist, and that sufficient bytes of the MAC would be pushed into the final block Cn, in order to sufficiently decrease the odds of a successful oracle response.

A 6 / n-6 record splitting approach seems to make intuitive sense, as it pushes 10 bytes of MAC into the final blocks of both split records. A record split would be executed at the browser above the encryption layer anytime that it receives a message length where length(message +MAC) % 16 ==0, i.e. a message that would result in a full block of padding. Upon identifying this condition, the browser splits the record into two records. The first record contains the first 6 bytes of the plaintext message, and the second record contains the remaining n-6 bytes. Both records are then independently MAC’d, padded, and encrypted. The effect is that the 20-byte MAC is pushed into the final message block, so that both messages contain 10 bytes of MAC, and 6 bytes of padding. This now decreases the probability that an attacker can replace the final block and achieve a successful match of both the final byte of padding AND 10 bytes of MAC in the last block from 1/256 to 1/288 = 3.23x10-27. This makes a successful attack much less likely. As seen below, the attacker is not successful after 3,000 oracle requests. Of course the ultimate solution is to disable SSLv3 altogether, as modern browsers have now done, but this record splitting may server as an effective interim solution. This defense approach could be further modified to split records with fewer than X MAC bytes in the final block, say if an attacker attempted to submit records with 15 bytes of padding and a single byte of MAC in the final block in an attempt to avoid a record split, if the cost of a 1/216 oracle success was still an affordable cost of effort.

One of the drawbacks to this defense is that it requires servers to correctly handle assembly of split records upon receipt, which has a non-trivial likelihood of causing interoperability problems between browsers employing a record-splitting defense and servers.







**References**

This POODLE Bites: Exploiting The SSL 3.0 Fallback, https://www.openssl.org/~bodo/ssl-poodle.pdf

POODLE attacks on SSLv3 (14 Oct 2014), Imperial Violet blog, https://www.imperialviolet.org/2014/10/14/poodle.html

[BEAST] T. Duong, J. Rizzo: “Here Come The ⊕ Ninjas”, 2011.