**CRIME & BREACH**

**CRIME**

[Compression Ratio Info-leak Made Easy (CRIME)](https://blog.qualys.com/product-tech/2012/09/14/crime-information-leakage-attack-against-ssltls) is a compression-based attack that targets TLS compression. As such, it can target sensitive information present in the HTTP body and HTTP headers such as session cookies. To successfully exploit CRIME, an attacker needs to be able to intercept traffic from the victim, as well as force the victim to adjust the request parameters slightly, for instance via malicious JavaScript code. The attacker also needs to know the name of the session cookie and the length of its value.

Let's look at an example to illustrate how the attack works. For our example we make the following assumptions:

* Let's assume the session cookie is called sess and has a length of 6 characters. The victim's session cookie's value is abcdef
* Our target website is called crime.local and we are attacking the path /crime.html
* A sliding-window compression algorithm is used that works similarly to LZ77 as discussed in the previous section

The attacker then forces the victim to request the target website but appends an extra HTTP parameter to the URL with the same name as the session cookie and an arbitrary value with the correct length. An exemplary request could look like this:

Code: http

GET /crime.html?sess=XXXXXX HTTP/1.1

Host: crime.local

Cookie: sess=abcdef

This request is now compressed using a sliding window compression algorithm, meaning that the sess= string present in the Cookie Header is replaced with a back-reference to the sess= string appended by the attacker to the query string. The compressed data is then encrypted. Since the attacker can intercept the ciphertext, he denotes the ciphertext size.

In the second step, the attacker changes the query parameter slightly to brute-force the value of the session cookie character by character from left to right. So, the next request might look like this:

Code: http

GET /crime.html?sess=aXXXXX HTTP/1.1

Host: crime.local

Cookie: sess=abcdef

In this case, the compression algorithm can now replace the string sess=a with a back-reference, since an additional character is the same in the cookie's value and query string. This means the resulting compressed data is smaller, potentially resulting in a smaller ciphertext. The attacker notices the smaller ciphertext and knows that the current character is correct. He can therefore move on to the next character:

Code: http

GET /crime.html?sess=aaXXXX HTTP/1.1

Host: crime.local

Cookie: sess=abcdef

The attacker can apply this technique recursively to brute-force all characters of the cookie, thereby leaking the session cookie. Depending on the length of the session cookie, a lot of requests are required to perform this attack.

**BREACH**

[Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext (BREACH)](https://breachattack.com/) is a variant of CRIME that targets HTTP-level compression. Since HTTP-level compression can only compress the HTTP body, BREACH is unable to target session cookies that are transmitted in HTTP headers. Therefore, potential targets of BREACH are sensitive information contained in the HTTP body such as CSRF-tokens.

Conceptually, BREACH works identically to CRIME with the slight difference that the webserver's response needs to contain a reflected value in the body for the attack to work since the attacker cannot simply adjust the query string as it is not part of the HTTP body.

**Tools & Prevention**

The simplest countermeasure to prevent CRIME attacks is to disable TLS-level compression. Alternatively, compression algorithms that do not fulfill the requirements needed for the successful exploitation of CRIME can be used to mitigate this attack. As of today, up-to-date webservers and libraries are not vulnerable to CRIME as patches have been applied.

Similarly, the simplest countermeasure to prevent BREACH attacks is to disable HTTP-level compression.