Mitigation of compression based attack on TLS/SSL

[1]Jateen Gadhiya, [2] Prof. Naimisha Trivedi
Computer Department
[1]L. D. College of Engineering, Ahmedabad, India
[1]jateen.gadhiya491@gmail.com, [2]naimishatrivedi@gmail.com

***Abstract*—Transport Layer Security(TLS) commonly known as Secure Socket Layer (SSL) is a foundation for Internet Security. We can achieve all security services by just implementing TLS. But though it reveals plaintext information through side channels. So, In this paper, we discuss security mechanism along with their side channel info-leakage. One basic side channel is compression, so based on compression information various attacks has been developed since last few years. Later, we discuss attacks like CRIME and BREACH briefly. In the end, we proposed schema to mitigate this attacks along with result analysis.**

***Keywords*— TLS, Attack, Security, Compression, Mitigations.**

I. INTRODUCTION

Every ecommerce or security oriented websites of today's internet are rely on Secure Socket Layer or TLS Encryption[7]. So, whenever we visit such website, our address in browser transform from 'http://' to 'https://'. Transport Layer Security is the best among possible security mechanism for internet security.

TLS provides Hash-Encode-Encrypt(HEE) mechanism to sensitive data transfer between client and server. TLS is bunch of three cryptographic algorithms by means of different security mechanism. Asymmetric encryption like RSA or Diffie-Hellman for authentication and negotiation, Integrity is made avail by generating hash using algorithm like MD-5 or SHA-1 and Symmetric Encryption to provide confidentiality of data by using strong ciphers like AES,RC4,DES etc[7]. So, breaking of such security takes years to decode encrypted information. So we can't crack it directly, but indirectly it reveals some plaintext information through side channel that we discuss later in this paper.

Later in this paper we discuss side channel role and then attacks developed based on that information. At end, we proposed scheme to overcome this vulnerability and mitigate such attacks and finally conclude with experimental results.

II. INFORMATION LEAKAGE

Cryptography is means to hide information and strength of encryption is manly depend on size of encryption key. For instance, if the key is 128 bit long then possible combinations for unique key is 2^{128} and break it through brute force takes 2^{24} years (if we consider 0.02sec per key) to try all possible combinations. So, practically break such ciphers using brute force is impossible.

However, this cryptographic foundation have some side channel as shown in Fig. 1. We can't hide them practically.

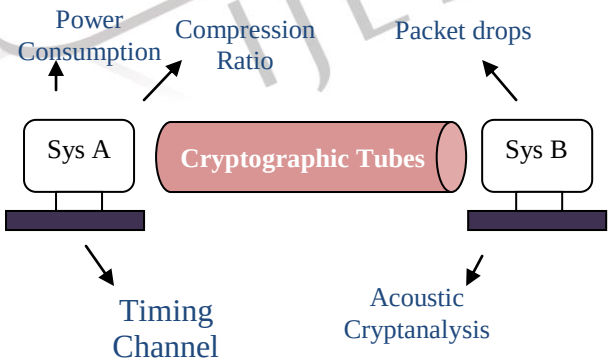

Figure 1 Side Channels in Cryptographic systems

As shown in Fig.1 that two systems A and B communicate over web have secure cryptographic protection as we discuss earlier. So, there is no chance for leakage of plaintext information from cryptographic tubes. But, some side channel information like HTTP compression ratio because of GZIP/DEFLATE[4], Timing data for request and response, encrypted packet size etc. These all side channels information may lead us to know something about plaintext information[6].

In this paper, we mainly focus on compression based side channel. There are two type of compression when we implement TLS/SSL: 1) TLS Compression (Optional) and 2) HTTP Compression. Compression give us knowledge of plaintext because

higher the redundancy in plaintext then there is more compression or higher compression ratio and vice versa. Now, we will discuss attacks developed based on this compression side channel with these both type of compressions.

III. ATTACKS BASED ON COMPRESSION

A. *CRIME*

CRIME stands for Compression Ratio Info-leak Made Easy[2]. CRIME is an attack on SSL/TLS that was developed by researchers Juliano Rizzo and Thai Duong. CRIME is a side-channel attack that can be used to discover session tokens or other secret information based on the compressed size of HTTP requests. The technique exploits web sessions protected by SSL/TLS when they use one of two data-compression schemes (DEFLATE and gzip [4]) which are built into the protocol and used for reducing network congestion or the loading time of web-pages.

B. *BREACH*

CRIME was further developed in 2013 into a hacking technique, dubbed as BREACH - Browser Reconnaissance & Ex-filtration via Adaptive Compression of Hypertext[1], that also exploits the use of data compression algorithms. A BREACH attack can extract login tokens, email addresses or other sensitive information from TLS encrypted web traffic in as little as 30 seconds (depending on the number of bytes to be extracted), provided the attacker tricks the victim into visiting a malicious web link or is able to inject content into valid pages the user is visiting (ex: a wireless network under the control of the attacker). All versions of TLS and SSL are at risk from BREACH regardless of the encryption algorithm or cipher used. Unlike CRIME, which can be successfully defended against by turning off TLS compression, BREACH exploits HTTP compression which cannot realistically be turned off, as virtually all web servers rely upon it to improve data transmission speeds for users. This is a known limitation of TLS as it is susceptible to chosen-plaintext attack against the application-layer data it was meant to protect. For more information on BREACH attack, visit official website http://www.breachattack.com[1].

IV. RESEARCH WORK

As we have studied CRIME and BREACH attacks, we found that side channels for TLS made it vulnerable. In latest update of TLS 1.2, they already made provision for mitigating CRIME attack by disabling TLS compression. This doesn't much affect the performance because it applied on encrypted text, but disabling HTTP compression is not feasible because it degrade the internet performance drastically, as it applied on human generated plaintext and we know that there more redundancy in human generated text then machine generated text. So, there is no concern for mitigating CRIME attack, but still there should be sufficient work to be done to mitigate BREACH attack. The designers of BREACH attack have already proposed some mitigations to overcome this vulnerability like[1]:

- Length Hiding
- Separating Secrets from User Input.
- Disabling Compression
- Masking Secrets
- Request Rate-Limiting and Monitoring

All the above proposed mitigations have not practically implemented and tested yet and everyone of them have their limitations. As disabling compression is feasible and secret separating required adequate knowledge of web application. Based on above mitigation we have proposed and implemented mitigations technique that would be discuss further in this paper and finally concluded with results.

A. *Proposed Mitigation*

It is not feasible to disabling the HTTP compression since every bit is charged over internet and by fact, DEFLATE[9] and Gzip reduce the web traffic up to 70%. Out of all the proposed mitigations, the better one and more efficient are length hiding and secret masking. Figure 2 shows proposed mitigation technique which is combination of both length hiding and secret masking, which may make BREACH attack impractical.

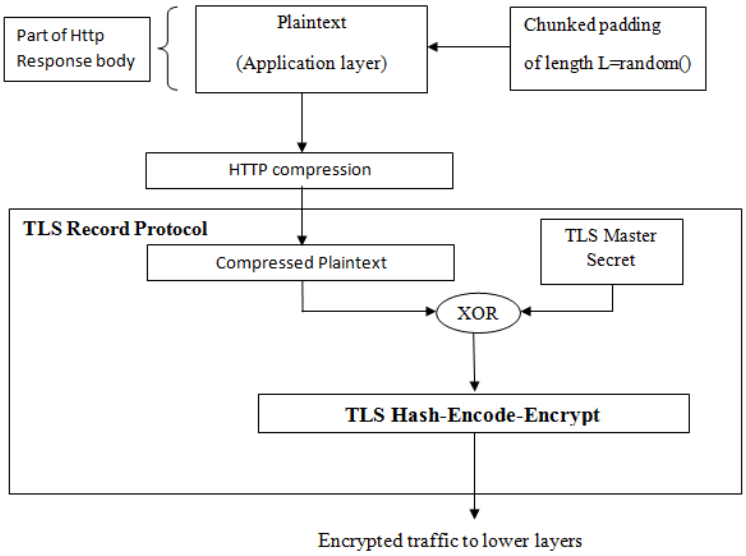

Figure 2 Proposed mitigation technique

As shown in Fig. 2, there are two appended mechanisms. First one is to append chunked padding data to original plaintext data generated by web server as part of HTTP response body. The chunked padding is variable length data. chunks carry some additional information, they affect the size of the response, but not the content. This make response length randomization. Response length randomization slows down the attacker, but does not prevent the attack entirely. But, if the attack can be slowed down significantly, perhaps it will be as good as prevented.

Chunked padding is followed by HTTP compression as normal operation. Then compressed data or plaintext is XORed with TLS master secret which is common to both client and server. This will mask the original data and master secret is identical to every TLS connection.

B. Experimental Setup

For implementing mitigation scheme over web server, we have to modified server architecture. But, for experiment purpose we have write PHP script that serve the same service that should be done by server. Also, this is not possible to execute proxy as well as server on same machine because it leads to port collision. As https enabled site and proxy.exe both require 443 port for execution. So, we have implemented attack setup in two computer connected through ad-hoc network. The architecture is shown in below figure 3.

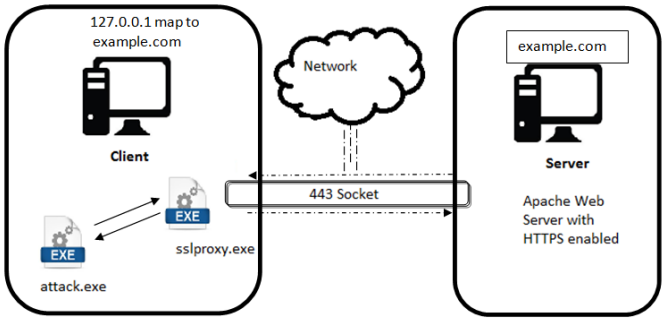

Figure 3 Exprimental setup

Under environment shown in figure 3, the BREACH attack is performed successfully. Time taken for complete and successful ex-filtration of secrets are purely based on number of bytes or size of secret and access or packet retrieval speed i.e internet speed.

C. Results

We had tested our implementation at various bandwidth of internet and with variable size of secrets to retrieve. We got the following results under the environment Windows 7 as Operating system, IIS 7 as Web server, OpenSSL for SSL implementation and modified attack binaries provided by authors.

At 2 Mbps network connection, results are:**(without proposed scheme)**

Sr No.	Secret Size (in Bytes)	Total Request Made	Retrieval time(sec)
1	8	547	187

2	16	819	250
3	32	1112	418

At 16 Mbps network connection, results are: **(without proposed scheme)**

Sr No.	Secret Size (in Bytes)	Total Request Made	Retrieval time(sec)
1	8	532	65
2	16	845	115
3	32	1102	147

At 2 Mbps network connection, results are:**(with proposed scheme)**

Sr No.	Secret Size (in Bytes)	Total Request Made	Retrieval time(sec)
1	8	1248	489
2	16	1612	600
3	32	2267	911

At 16 Mbps network connection, results are: **(with proposed scheme)**

Sr No.	Secret Size (in Bytes)	Total Request Made	Retrieval time(sec)
1	8	1202	112
2	16	1570	210
3	32	2190	345

From the above data, we have noticed that after implementing our scheme, the number of requests made to retrieve secret is increased very much. Thus it increase the retrival time too. Also we have to mentioned that during execution, somtimes it gave wrong secret output because of same priority byte collision.

V. CONCLUSIONS

Thus from our results we conclude that proposed mitigation technique works effectively and it block away attacker for more time to retrieve sensitive data. But, this technique doesn't fully mitigate the attack because attacker can develop logic based on taking average of multiple response size and able to guess the correct byte. This increase the execution time but not full mitigate the attack.

In future, better mitigation technique can be developed and also, better attack scenario can be made because it have some technical challenges to overcome (refer official attack paper [1]).

REFERENCES

[1] Gluck, Yoel, Neal Harris, and Angelo Ángel Prado. "Breach: Reviving The Crime Attack." (2013).

[2] Rizzo, Juliano, and Thai Duong. "The CRIME attack." *ekoparty Security Conference*. Vol. 8. 2012.

[3] Rizzo, Juliano, and Thai Duong. "Practical padding oracle attacks."*Proceedings of the 4th USENIX conference on Offensive technologies, WOOT*. Vol. 10. 2010.

[4] Deutsch, L. Peter. "DEFLATE compressed data format specification version 1.3." (1996).

[5] As of October 02, 2013. "SSL Pulse: Survey of the SSL Implementation of the Most Popular Web Sites". Retrieved 2013-10-10.

[6] Kelsey, John. "Compression and information leakage of plaintext." *Fast Software Encryption*. Springer Berlin Heidelberg, 2002.

[7] Wikipedia contributors. "Transport Layer Security." *Wikipedia, The Free Encyclopedia*. Wikipedia, The Free Encyclopedia, 4 Dec. 2013. Web. 10 Dec. 2013.

[8] Corella, Francisco, and Karen Lewison. "It Is Time to Redesign Transport Layer Security." (2013).