iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 6 **Issue:** VII **Month of publication:** July 2018

DOI: http://doi.org/10.22214/ijraset.2018.7145

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887
Volume 6 Issue VII, July 2018- Available at www.ijraset.com

Analysis and Counter Measures of Various Attcks in Secure Network Protocols

Dr. V. Umadevi [1], P.Thanapalan[2]
[1]*Research Director, Department of Computer Science, Jairams Arts and Science College, Karur.*
[2] *Research Scholar, Department of Computer Science, Jairams Arts and Science College, Karur.*

Abstract: System Protocols are basic to the activity of the Internet and henceforth the security of these conventions is principal. Our work covers the security of three generally sent conventions: Domain Name System (DNS), Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS). Our work demonstrates that the outline or usage of a few variations of these conventions are defenseless against assaults that come guarantee their key security highlights. In the greater part of the cases we incorporate exploratory outcomes showing the practicality of our assaults in reasonable system conditions. We propose various countermeasures for the assaults, some of which have just been executed by and by. We begin by depicting the structure of DNS and present various existing DNS security conventions. We at that point center around DepenDNS, a security convention that is proposed to ensure DNS customers against store harming assaults. We show that DepenDNS suffers from operational deficiencies, and is powerless against reserve harming and disavowal of administration assaults. We at that point give a review of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS), and draw the likenesses and differences between the two conventions. We depict the cushioning prophet idea and present various ongoing assaults against TLS.

I. INTRODUCTION

The evolution of secure network protocols has been largely driven by the discovery and the successful exploitation of weaknesses in either the design or the implementation of these protocols. The Domain Name System (DNS) and Transport Layer Security (TLS) provide good examples that demonstrate this broken reactive model of evolution.

Maintaining the security of DNS has been a continues challenge with high-pro le and high-impact attacks frequently emerging (for example, Kaminsky's cache poisoning attack against DNS), which are usually followed by the development of ad hoc security protocols that try to protect DNS from these attacks. In most cases, attacks against DNS have exploited trivial, and on occasion known, weaknesses. Most of these attacks would have been prevented if the basic DNS security mechanisms had been deployed.

TLS, on the other hand, is by far the most widely deployed secure network protocol today, and which best show-cases the failure of this ad hoc approach of designing and implementing secure network protocols. Attacks of di erent severity and practicality levels have been published against TLS (and its predecessor, Secure Socket Layer), triggering ad hoc and non-coordinated responses from the Internet Engineering Task Force (IETF) who maintain the protocol speci cation, and the TLS open and closed source code development community. Alarmingly, the number of attacks against TLS has recently been on the rise including, for example, BEAST, CRIME, and BREACH. Our work takes advantage of previously unknown weaknesses introduced by this ad hoc approach to develop attacks that exploit the above mentioned protocols using basic, but novel, techniques.

II. SIDE CHANNEL ATTACKS

Physical attacks on cryptosystems take advantage of implementation-speci c charac-teristics to recover secret parameters that are involved in di erent computations. An attack that exploits information leaked by a system is generally referred to as a side channel attack; typically, exploiting unintended leakage that hardware and software implementations of cryptosystems may produce. Sources of hardware leakage include timing [1, 2], power consumption [3] and electromagnetic radiation [4]. Sources of software leakage include error messages [5] and message sizes [6]. An attacker exploiting a side channel can gain further information that may potentially help him exploit a cryptosystem. For example, this might help an attacker recover the encryption key or recover encrypted messages.

III. TIMING SIDE CHANNEL ATTACKS

A timing side channel attack is, essentially, a way of obtaining secret information by measuring the time it takes for cryptographic operations to complete. The timing vari-ance in the operations reveals enough information to the attacker for him to acquire relevant, and possibly sensitive, information about the system. Timing side channel attacks are typically coupled with statistical

analysis. The timing side channel attack idea was rst introduced by Kocher in [1], where he showed that carefully measur-ing the amount of time required to perform private key operations could possibly help attackers nd xed Di e-Hellman exponents, factor RSA keys, and break other cryp-tosystems. The fundamental idea is taking advantage of timing variations. The idea was then translated into a real attack against a smart card-based implementation of RSA [3]. Boneh and Brumley later took this further and demonstrated in how to extract private keys from an OpenSSL-based web server running on a machine in the local network.

A. Padding Oracles

An oracle can be thought of as a black box that responds to queries. A padding oracle reveals side channel information that indicates the correctness of padding. In certain circumstances, a padding oracle can be leveraged to build a decryption oracle, that is, enables plaintext attacks against a protocol. For it to be useful to an adversary, a padding oracle must have a practical realisation. This realisation can be achieved by exploiting the leakage of padding-related side channel information such as error messages or timing.

The concept of a padding oracle was rst introduced by Vaudenay [7]. In Vaudenay's formulation, a padding oracle is a notional algorithm which, when presented with a CBC-mode ciphertext, returns VALID if the underlying plaintext has padding that is correctly formatted and INVALID otherwise. Here, correctness is with respect to some padding scheme. For example, for (D)TLS padding, correctness means that the decryption of the ciphertext is a byte string ending in one of the valid padding pat-terns \0x00", \0x01 0x01", etc. Vaudenay showed that, for certain padding schemes, repeated access to a padding oracle can be used to decrypt arbitrary target ciphertext blocks (and indeed complete ciphertexts in a block-by-block manner).

The Canvel et al. realisation of the padding oracle relies on the fact that, for a TLS implementation, the processing of a message with valid padding may take longer than the processing of a message with invalid padding. The reason for this is that the padding is checked for validity before the MAC verification is performed, and so a TLS implementation that aborts processing immediately after detecting an error (of any kind) will exhibit a timing difference in message processing for packets with valid and invalid padding: in the former case, the MAC verification will take place, while in the latter it will not. The timing di erence would then show up as a di fference in the time at which the error messages appear on the network. As observed in [25], this is exactly how TLS was implemented in OpenSSL.

IV. THE BEAST ATTACK

The so-called BEAST attack [9] is a blockwise-adaptive chosen-plaintext attack (BACPA) that can be mounted against SSL and TLS 1.0. This attack exploits the use of chained initialisation vectors (IVs) for CBC-mode and has its roots in [10]. The BEAST attack achieved full plaintext recovery against SSL and TLS 1.0, but only in scenarios where an attacker can gain access to a chosen plaintext capability, perhaps by inducing the user to rst download malicious JavaScript code into his browser. It is worth noting that the attack presented in [9] was constrained to the web setting, i.e. where TLS communications take place between a browser and a web server. In [9], it is assumed that the attacker has network eavesdropping, chosen-boundary and blockwise privileges. The chosen-boundary privilege allows the attacker to control block boundaries by prepending variable length sequences of bytes to a record, while the blockwise privilege allows him to prepend plaintext blocks to ongoing requests [10], i.e. the attacker can insert his block as the rst block for encryption. The network eavesdropping privilege provides the attacker with access to O_{BA}, an oracle that returns TRUE when two ciphertext blocks match and FALSE otherwise. The reader will nd that our description of the attack and the symbols we use slightly differ from [10]. Our goal is to give the reader a clear description of the attack that could be easily implemented. We first describe the attack model, list a number of assumptions and discuss how to implement the three privileges described above. We then describe the steps that the attack takes to recover an unknown message, one byte at a time.

V. OTHER ATTACKS AGAINST TLS

A. Distinguishing Attack Against TLS with Short MACs

The authors of [11] described a distinguishing attack against the MEE-TLS-CBC con-struction. Their attack exploits the use of short MACs in TLS as standardised in RFC 6066 and TLS's support for variable length padding. The outline of their attack is that if the size of the MAC is smaller than the size of the cipher block, and the plaintext message is small enough, then a distinguishing attack against TLS, with the MEE-TLS-CBC construction, can be mounted. The authors of [11] described how to distinguish whether an encrypted message contains for example YES or NO by modifying a few bits in the original ciphertext, C. The response (or lack of response) from the receiver of the TLS record helps the attacker identify whether the original message was YES or NO. The authors of [11] argue that the attack can be mounted in practice against TLS; they refer to the use of 80-bit truncated MACs in extensions to TLS 1.2, de ned in RFC 6066 .

B. The CRIME Attack

We described how TLS supports optional compression. The so-called CRIME attack was published in 2012 by the same authors of the BEAST attack [9]. The attack exploits the use of the DEFLATE compression method, imple-mented by the TLS Record Protocol and negotiated during the Handshake Protocol, in combination with a chosen plaintext capability to mount a plaintext recovery attack. The attack exploits side-channel information in the form of message size that is leaked when crafting web requests using a malicious web agent such as a JavaScript. All ver-sions of TLS (and SSL) were vulnerable to the CRIME attack, regardless of their mode of operation

C. The BREACH Attack

The so-called BREACH attack [12] con rms the statement we made in the introduction chapter of the thesis that the interaction of secure network protocols with their upper and lower-layers plays a critical part in defining the system's overall security. Although implementations of TLS disabled the use of compression after the CRIME attack, as discussed earlier, the authors of the BREACH attack demonstrated how to exploit HTTP-level compression to mount a chosen plaintext attack that can recover TLS-protected plaintext. The BREACH attack relies on exploiting side channel information leaked in HTTP responses rather than requests. As expected, the attack applies to all versions of TLS with all modes of operation.

D. RC4 Attack

The authors of [15] present ciphertext-only plaintext recovery attacks against TLS when RC4 is selected for encryption. The authors of [15] identi ed new biases in the RC4 key stream output. In the multi-session setting, these biases can be used to recover plaintext with varying probability of success, depending on the position of the byte to recover and the amount of ciphertext captured. The authors of [15] also demonstrate how to exploit biases in consecutive pairs of bytes in the RC4 keystream that were rst reported by Fluhrer and McGrew [13].

E. Attacks Against DTLS

DTLS has not been put under as much scrutiny as TLS. This is largely attributable to the protocol's recent introduction and its limited, but growing, number of implemen-tations, when compared to TLS. Most of the identi ed security issues with DTLS were associated with the protocol's implementation, with most of these issues resulting in a form of denial of service.

We have identified a number of security issues in one of Cisco Systems' DTLS implementations, which was based on OpenSSL. We successfully conducted denial of service and cipher suite downgrade attacks against the ASA line of products, in which DTLS is used to provide a re-mote access VPN. In fact, during this work we identi ed a critical OpenSSL software vulnerability that was independently discovered by another researcher and reported as CVE-2010-4180. We communicated, privately, our ndings to the vendor and worked with them to implement and test appropriate xes.

We argue that our attacks against DTLS, which we present in later chapters, are by far the most involved and high-impact work carried out against the DTLS protocol, to date. In fact, performing a basic Internet web search for \DTLS vulnerability" reveals links mostly pointing to our work against DTLS

TABLE 1
COUNTERMEASURE OF VARIOUS ATTACKS

Sno	Attacks	Countermeasure
1	Canvel et al. Timing Attack Against TLS	The attack of Canvel et al. was perceived as serious enough that the OpenSSL code for TLS was updated from releases 0.9.6i and 0.9.7a in an attempt to ensure that the processing time for TLS messages is essentially the same, whether or not the padding is correct, and to send the same encrypted error message, bad record mac, in both cases. Eventually, the same countermeasures appeared in the speci cation for TLS 1.1 [14], with the requirement that they must be implemented.
2	The BEAST Attack	The author of [10] suggested a number of countermeasures to defeat BACPA, well before the BEAST attack was released:

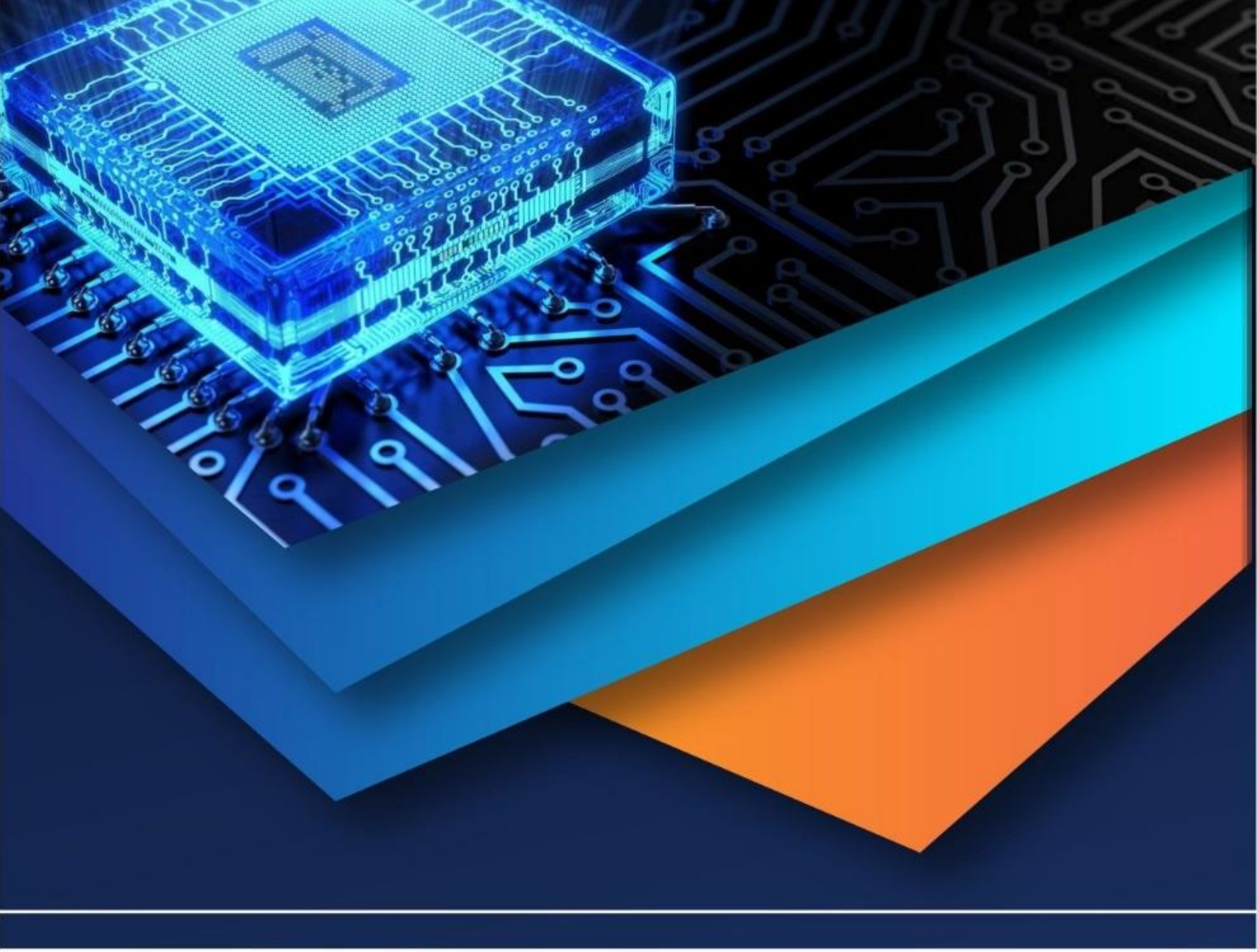
		Upgrade to TLS 1.1 (or TLS 1.2) in which explicit IV s are used. Keep using TLS 1.0, but introduce a single dummy rst plaintext block in every TLS record. This dummy block can be for example an all-zero string. Keep using TLS 1.0, but send an empty message that has no data, but that would still result in adding only padding and MAC, i.e. the CBC-encrypted part of such a record will consist just of a MAC and padding.
3	Distinguishing Attack Against TLS with Short MACs	The attack was considered to be more theoretical since no short MAC algorithms were supported in implementations of TLS. In addition, the work in [11] was not extended to a plaintext recovery attack. The recommended countermeasure would be not to use truncated MACs with the MEE-TLS-CBC construction.
4	The CRIME Attack	The workaround that was suggested and eventually deployed by most TLS implemen-tations was to disable the use of TLS compression, i.e. implementations of TLS must make sure that the use of compression is not o ered by the client in the Handshake Protocol's ClientHello message or is ignored by the server in case it was offered.
5	The BREACH Attack	A number of countermeasures are suggested in [12]. Examples of countermeasures listed in include disabling HTTP compression, length hiding and limiting the number of cookie requests.
6	RC4 Attack	A number of countermeasures have been proposed in [14]. It is worth noting that the attacks in require large amounts of ciphertext and hence their practical relevance could be questioned. Despite this, countermeasures were implemented in practice to defeat the attacks. For example, Opera has implemented a cookie limiting counter-measure, while Microsoft has modified their code so that RC4 is no longer enabled by default for TLS in Windows 8.1 Preview

VI. CONCLUSION

We covered a number of attacks against TLS that are relevant to the work we present in the next two chapters. The attacks we described in this chapter demon-strate the fact that not addressing what are considered at one point of time theoretical weaknesses could eventually lead to attacks that are practical and serious against a protocol, requiring ad hoc industry reaction (the BEAST attack being an example). As security researchers, we believe that \attacks only get better". Despite the number of high-pro le attacks against the TLS MAC-then-Encode-then-Encrypt construction, and recently against RC4, we are yet to see a significant uptake of TLS 1.2 and authenticated encryption algorithms. Authenticated encryption algorithms deliver the two functions simultaneously: data encryption and authentication.

REFERENCES

[1] B. K.opf and D. Basin. An Information-Theoretic Model for Adaptive Side-Channel Attacks. In Proceedings of the 14th ACM conference on Computer and communications security, pages 286{296. ACM, 2007
[2] D. Brumley and D. Boneh. Remote Timing Attacks are Practical. Computer Networks, 48(5):701{716, 2005.
[3] P. Kocher, J. Ja e, and B. Jun. Di erential Power Analysis. In Advances in Cryptology { CRYPTO'99, pages 388{397. Springer, 1999
[4] B. Koopf and D. Basin. An Information-Theoretic Model for Adaptive Side-Channel Attacks. In Proceedings of the 14th ACM conference on Computer and communications security, pages 286-296. ACM, 2007
[5] J. P. Degabriele and K. G. Paterson. Attacking the IPsec Standards in Encryption-only Con gurations. In IEEE Symposium on Security and Privacy, pages 335{349. IEEE Computer Society, 2007
[6] Alfredo Pironti and Pierre-Yves Strub and Karthikeyan Bhargavan. Identifying Website Users by TLS Tra c Analysis: New Attacks and E ective Countermeasures. Technical Report 8067, INRIA, September 2012
[7] S. Vaudenay. Security Flaws Induced by CBC Padding Applications to SSL, IPSEC, WTLS... In Advances in Cryptology EUROCRYPT 2002. Springer, 2002.
[8] B. Canvel, A. P. Hiltgen, S. Vaudenay, and M. Vuagnoux. Password Interception in a SSL/TLS Channel. In D. Boneh, editor, CRYPTO, volume 2729 of Lecture Notes in Computer Science, pages 583{599. Springer, 2003
[9] T. Duong and J. Rizzo. Here come the Ninjas. Unpublished manuscript, 2011.
[10] G. V. Bard. A Challenging but Feasible Blockwise-Adaptive Chosen-Plaintext Attack on SSL. In SECRYPT, pages 99{109, 2006
[11] K. G. Paterson, T. Ristenpart, and T. Shrimpton. Tag Size Does Matter: Attacks and Proofs for the TLS Record Protocol. In D. H. Lee and X. Wang, editors, ASIACRYPT, volume 7073 of Lecture Notes in Computer Science, pages 372-389. Springer, 2011.
[12] Y. Gluck, N. Harris, and A. Prado. BREACH: Reviving The Crime Attack. http://breachattack.com.
[13] S. R. Fluhrer and D. A. McGrew. Statistical Analysis of the Alleged RC4 Keystream Generator. In FSE, pages 19{30, 2000.
[14] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346, Internet Engineering Task Force, Apr. 2006.
[15] N. AlFardan, D. J. Bernstein, K. G. Paterson, B. Poettering, and J. Schuldt. On the Security of RC4 in TLS and WPA. In USENIX Security Symposium, 2013



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY