Università del Piemonte Orientale

Dipartimento di Scienze e Innovazione Tecnologica

Corso di Laurea in Informatica

Relazione per la prova finale

ATTACCHI AL PROTOCOLLO MMS

Tutore interno:

Prof.ssa Lavinia Egidi

Candidato:

Andrea Ierardi

Anno Accademico 2018/2019

Elenco delle figure

Indice dei contenuti

Abstract

In questa relazione viene principalmente discusso il problema della protezione dei dati inviati da client e server che utilizzano il protocollo di comunicazione MMS con crittografia TLS. Quest'ultima non garantisce una sicurezza completa e perfetta dei dati poiché molti sistemi informatici non sono aggiornati a versioni qualitativamente migliori e presentano parecchie vulnerabilità. Uno studio di queste vulnerabilità ha portato alla realizzazione di una serie di attacchi mirati: Sniffing, ARP poisoning, attacco Man in the Middle, attacco Denial of Service, attacco Packet Filtering, attacco Downgrade che però non ha avuto successo. Per la loro realizzazione sono stati utilizzati tool, script e software open-source. Da questo studio si può dedurre come sia facile per un utente esperto, eludere le difese di alcuni dispositivi datati o non aggiornati. I sistemi che utilizzano protocolli obsoleti sono in maggioranza e inoltre, parecchi utenti tendono ad ignorare l'importanza della sicurezza che attualmente sta assumendo un ruolo chiave nel mondo dell'informatica.

1. Introduzione

Il tema della sicurezza informatica è uno dei più trattati e in voga del momento. In questi ultimi anni i criminali, i malware e gli attacchi informatici sono incrementati. Attualmente, i sistemi sono sempre più a rischio poiché ogni giorno vengono scoperti nuovi tipi di attacco e vulnerabilità. Nel 2018 è stata rilasciata una nuova versione del protocollo Transport Layer Security (TLS), ovvero TLS1.3. Questo protocollo risolve molte delle vulnerabilità introdotte con il suo predecessore TLS1.2, che non è più ritenuto sicuro. I dispositivi però non sono stati al passo con questo cambiamento e molti supportano ancora versioni obsolete e non sicure. Inoltre, con l'avanzamento verso l'industria 4.0 e la nascita di dispositivi sempre più interconnessi, la privacy e la sicurezza dei dati sta acquisendo una grande importanza. La crescita della quantità di dati è sempre maggiore e sono necessarie tecniche migliori e all'avanguardia. Uno standard parecchio diffuso nell'industria Operational Technology (OT), in particolare per l'operatività del sistema elettrico, è il protocollo Manufacturing Message Specification (MMS).

In questa relazione verranno presentati alcuni attacchi contro questo protocollo con supporto alla crittografia TLS. Verranno presentati nello specifico gli strumenti e le strategie utilizzati per la loro realizzazione. Inoltre, verrà anche riportato uno studio sulle principali vulnerabilità di TLS fino alla versione 1.2.

2. Scopo del lavoro

Lo scopo principale del lavoro è stata la realizzazione di tool di attacco contro il protocollo MMS, con supporto alla comunicazione sicura. Una parte del progetto è stata di analisi e di ricerca sulle vulnerabilità di TLS e dei vari cipher suite.

Ricerca sul Sistema Energetico (RSE) [1] è un'azienda che svolge attività di ricerca nel settore elettro-energetico, progetti di impresa, prove di laboratorio per effettuare verifiche sperimentali, test su strumenti di misura e su aspetti di cybersecurity. Ho svolto uno studio di 250 ore in università, guidato dal mio tutore interno nell'ambito di una collaborazione scientifica dell'Ateneo con RSE. Il lavoro si inquadra in un'attività di realizzazione di un laboratorio per la simulazione e l'analisi di attacchi al sistema power. Questo tirocinio mi ha permesso di espandere le mie conoscenze nel campo della sicurezza informatica soprattutto dal punto di vista sperimentale. Ho lavorato al progetto in modo autonomo e ho avuto un certo grado di libertà dal punto di vista della scelta degli strumenti. L'unico limite posto è stato l'utilizzo dei container Docker per la simulazione degli attacchi, del client e del server MMS. Grazie alle funzionalità di Docker è stato possibile implementare ambienti di simulazione semplici, adeguati e veloci.

3. Descrizione delle tecnologie convolte

3.1 La struttura di Docker e dei containers

3.1.1 Introduzione a Docker

Docker [2] è un software open-source che permette l'automatizzazione della distribuzione di applicazioni in contenitori portabili e autosufficienti che possono essere eseguiti nel cloud o in locale. E' anche possibile creare reti che permettono di interconnettere gli host tra di loro. La struttura consiste in applicazioni incapsulate in container (si veda la Figura 1) che possono dialogare tra loro grazie alla rete Docker. Questi possono essere applicativi o sistemi operativi a sé stanti. Uno dei vantaggi principali di questa soluzione, è la portabilità, infatti, è possibile eseguire i container in qualunque sistema operativo, senza problemi di compatibilità.

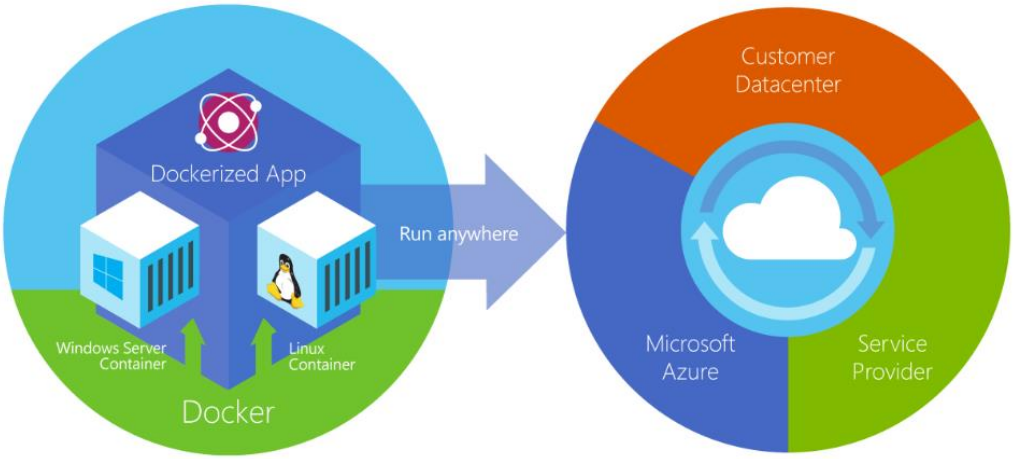

Figura 1: La struttura dei container Docker [2]

3.1.2 I container Docker

I containers sono delle strutture che permettono la simulazione di applicativi, servizi e sistemi operativi senza dover ricorrere ad alcuna macchina virtuale; bensì avviene tutto all'interno dello stesso Kernel. In questo modo, la struttura rimane compatta, semplice e veloce (si veda Figura 2). Non avere una macchina virtuale garantisce vari vantaggi: portabilità, velocità di esecuzione e anche un risparmio di tempo durante l'avvio. Per avviare un container sono necessari pochi secondi, mentre per la creazione di una macchina virtuale è necessario dedicare del tempo per tutte le impostazioni relative all'immagine da simulare. I container Docker risultano più pratici rispetto alla virtualizzazione.

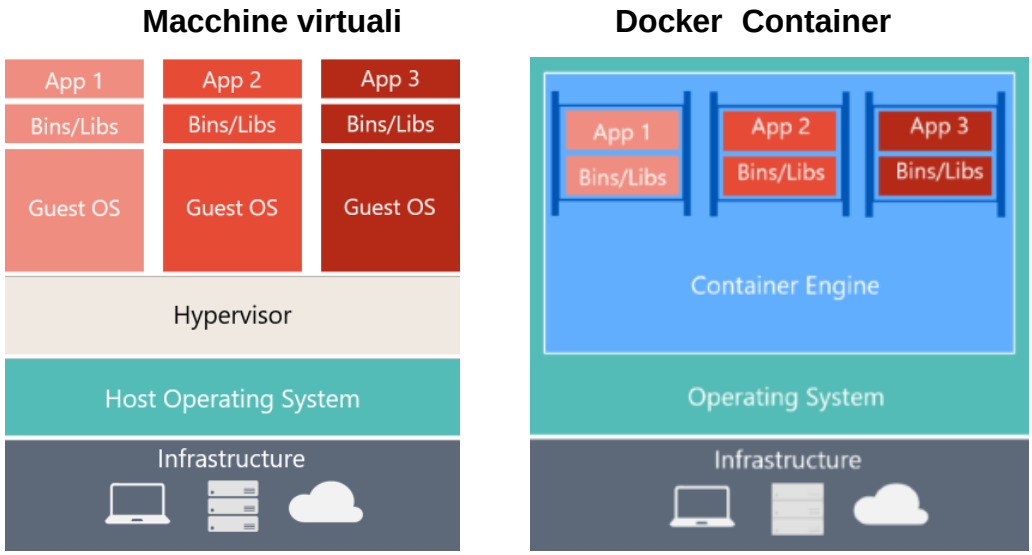

Figura 2: Macchine virtuali tradizionali vs. Container Docker [2]

3.1.3 Le reti Docker

Una delle funzioni principali di Docker è la possibilità di definire delle reti [3] tra i container (si veda Figura 3). Queste permettono di simulare un ambiente di comunicazione reale tra host differenti, nel quale ognuno offre servizi o simula sistemi operativi. Inoltre, è possibile la modifica di una rete già esistente oppure la costruzione di una ad hoc. Quest'ultima è detta *user-bridge network* [4] e consente all'utente di poter gestire tramite la modifica di un file JSON: il nome della rete, i container che possono aderire, gli indirizzi Internet Protocol (IP) del gateway e del Domain Name Service (DNS) di default, la Maximum Transmission Unit (MTU) dei pacchetti e altre informazioni. [5]

Quando un container viene portato dallo stato di build allo stato di running si connette in automatico alla rete di default che è detta *default-bridge network* [4]. Quest'ultima ha impostazioni di default ed è la più semplice da configurare dato che all'avvio di un container, Docker ne imposta in automatico l'indirizzo IP e altre impostazioni di rete.

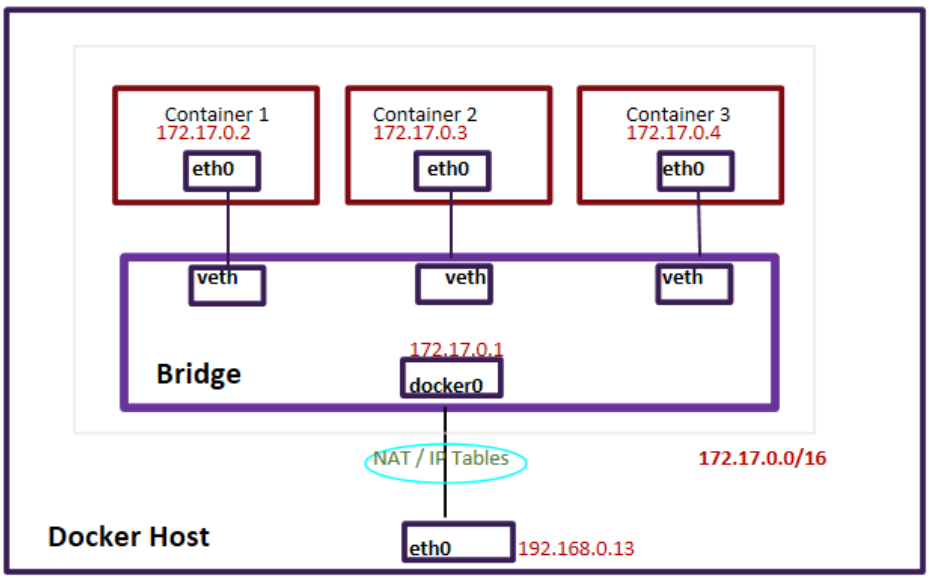

Figura 3: Docker Bridge network [6]

3.2 I protocolli Secure Socket Layer e Transport Layer Security

3.2.1 Introduzione

TLS [7] e la sua versione precedente, Secure Socket Layer (SSL), sono dei protocolli crittografici a livello presentazione del modello International Organization for Standardization/Open Systems Interconnection (ISO/OSI) e vengono utilizzati per l'implementazione di una comunicazione sicura tra due o più host. Dalla sorgente al destinarlo forniscono autenticazione, integrità dei dati e riservatezza operando al di sopra del livello di trasporto. Il protocollo SSL/TLS può essere diviso in due layer: handshake layer e record layer. Client e server, durante la comunicazione, devono accordarsi sull'utilizzo di TLS, instaurando una connessione attraverso la procedura di handshake. Questa è definita da una serie di messaggi che permettono la negoziazione di parametri di sicurezza di una sessione. Il Record Protocol prende i dati, li frammenta in modo tale da renderli adeguati per l'algoritmo crittografico, opzionalmente li comprime, applica un *message authentication code* (MAC) o un *keyed-hash message authentication code* (HMAC) e infine cifra o decifra i dati usando le informazioni negoziate durante il protocollo di handshake.

3.2.2 Fasi dell'handshake TLS/SSL

Le fasi dell'handshake TLS/SSL comprendono una fase di negoziazione [8] (si veda Figura 4):

1. Il client invia un messaggio di ClientHello dove specifica la migliore versione del protocollo TLS supportata, un numero random e la lista dei cipher suite suggeriti.
2. Il server risponde con un messaggio di ServerHello che contiene la versione del protocollo scelta, un numero random, il cipher suite e il metodo di compressione tra quelli offerti dal client. Inoltre, manda il proprio certificato e ne richiede uno anche al client se è specificata l'autenticazione.
3. Il client verifica il certificato del server.

4. Il client manda una stringa di byte random che permette ad entrambi il calcolo della chiave segreta da usare per cifrare i messaggi successivi. Questi random byte sono cifrati con la chiave pubblica del server.
5. Se il server ha richiesto il certificato del client, quest'ultimo invia una stringa di byte random cifrati con la propria chiave privata insieme al certificato.
6. Il server verifica il certificato del client.
7. Il client manda un messaggio di terminazione, che viene cifrato con la chiave segreta, indicando che la parte del client nell'handshake è stata completata.
8. Il server invia un messaggio di terminazione che è cifrato con la chiave segreta, indicando che la parte del server nell'handshake è stata completata.
9. Per il mantenimento della sessione SSL/TLS, il server e il client ora possono scambiarsi messaggi che sono simmetricamente cifrati con la chiave segreta condivisa.

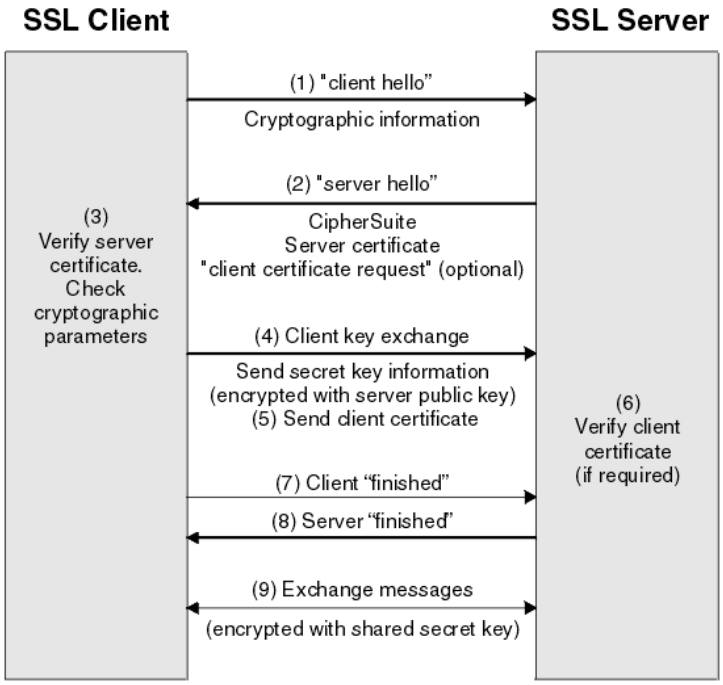

Figura 4: Una panoramica dell'handshake TLS/SSL [8]

3.3 Protocollo Manufacturing Message Specification

3.3.1 Descrizione del protocollo MMS

Il protocollo MMS [9] è un sistema internazionale standardizzato per lo scambio di messaggi supervisionati in tempo reale e di informazioni di controllo tra i dispositivi e computer di rete. L'invio dei messaggi fornito da MMS è generico e questo lo rende conforme con gli standard di una grande varietà di dispositivi, applicazioni e industrie. MMS offre un grande numero di servizi come la lettura o scrittura di variabili, download e upload di programmi e il controllo dell'esecuzione di programmi da remoto.

I principali vantaggi derivati dall'utilizzo di questo protocollo sono [10]:

- **Interoperabilità**: è l'abilita di due o più applicazioni di rete di scambiare controlli supervisionati e dati sulle informazioni dei processi senza che l'amministratore dell'applicazione debba creare un ambiente di comunicazione.
- **Indipendenza**: l'interoperabilità rende l'applicazione indipendente dallo sviluppatore che non deve agire direttamente sul protocollo.

3.3.2 Il modello a oggetti di MMS

Il modello a oggetti [9] ha ottenuto una grande accettazione nei campi dell'analisi e del design di sistemi complessi. Essenzialmente, un oggetto è la rappresentazione di un'entità del mondo reale ed è caratterizzata da attributi e operazioni che agiscono su tali attributi. Le operazioni principali sugli oggetti sono chiamati servizi, i quali vengono forniti dal server. Questi vengono utilizzati principalmente per poter accedere e modificare gli attributi. L'idea è di definire modelli di oggetti per device e applicazioni che siano sufficientemente astratti da nascondere gli aspetti di implementazione (si veda Figura 5). In questo modo l'applicazione non accede al dispositivo reale ma all'oggetto MMS che lo rappresenta. Questo è sufficiente per definire quasi tutti gli elementi di un sistema industriale.

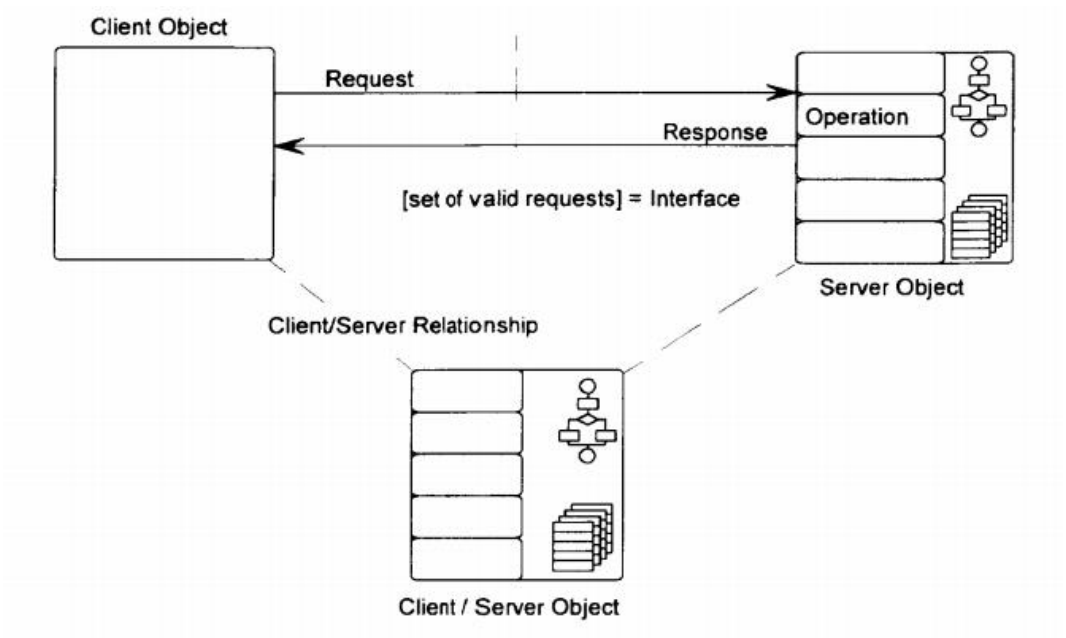

Figura 5: Il modello a oggetti di MMS [9]

3.3.3 Server MMS della libreria libIEC61850

La libreria libIEC61850 [11] fornisce un application programming interface (API) ad alto livello e specifica per lo standard IEC 61850. Questa include la generazione automatica dei modelli di dati e dei device del protocollo MMS. Inoltre, fornisce un supporto per i controlli, i servizi di logging e varie impostazioni. Il server utilizza di default la porta 3782 per la comunicazione sicura, mentre è possibile specificare la porta per la comunicazione in chiaro. Il programma che appartiene alla libreria libIEC61850 e preso in esame in questo studio è il file *tls_server_example.c*. Questo server instaura una connessione sicura nella quale vengono inviati i valori di parametri che vengono richiesti da comandi generici o di reporting. La versione di TLS utilizzata per la comunicazione è la versione TLS1.1 e la scelta del cipher suite nella fase di server hello è RSA con AES256, CBC e SHA.

3.3.2 Client MMS della libreria libIEC61850

Il client [11] supporta lettura e scrittura di variabili, reporting e controllo dei servizi. Per avviare una comunicazione con un altro host, l'utente può specificare al client l'indirizzo IP e la porta al quale connettersi.

Il programma che appartiene alla libreria libIEC61850 e utilizzato per i test in questo studio è il file *tls_client_example.c*. Questo client comunica in modo sicuro una serie di comandi al server che risponde con i parametri richiesti. La versione di TLS utilizzata è la 1.1 con cipher suite RSA con AES256, CBC e SHA ed è quella che verrà concordata con il server durante l'handshake TLS.

4. Lavoro svolto e soluzioni proposte

4.1 Ricerca sulle vulnerabilità dei protocolli e dei cipher suites

4.2.1 Introduzione

Negli ultimi anni la sicurezza dei protocolli SSL/TLS è stata continuamente minacciata dalla nascita di nuovi ed efficaci attacchi informatici. Lo sviluppo e l'inventiva degli attaccanti è ogni giorno in continua crescita, mentre sempre più dispositivi supportano protocolli datati, non più considerabili sicuri. Questo accade in parecchi uffici pubblici, ospedali e aziende private.

4.2.1 Vulnerabilità concettuali in TLS e attacchi risultanti

Alcune delle vulnerabilità del protocollo TLS sono concettuali dello standard, ovvero vengono sfruttati errori derivati da meccaniche fondamentali del protocollo come la fase di handshake. Molte di queste comprendono attacchi di downgrade, rinegoziazione della connessione e continuazione di sessione.

Alcuni di questi attacchi sono [12]:

- *3SHAKE* richiede che il client si connetta ad un server infetto e presenti le proprie credenziali in modo che l'attaccante possa impersonare il client a qualsiasi altro server che accetti queste credenziali. Questo attacco funziona contro server che utilizzano un'autenticazione basata sul certificato e che supportano la rinegoziazione. Per mitigare questo attacco è necessario l'utilizzo di TLS1.3 che non permette la rinegoziazione.

- *Padding Oracle On Downgraded Legacy Encryption* (POODLE) è un attacco man in the middle che esegue un downgrade attack ai protocolli TLS.1.0, 1.1 o 1.2 a SSLv3.0 per l'esecuzione di un attacco a forza bruta contro il padding di CBC.

- *LOGJAM* utilizza l'attacco downgrade contro connessioni TLS che usano una crittografia di esportazione di 512 bit e che utilizza gruppi Diffie-Hellman deboli. Questo permette all'attaccante di leggere e modificare qualsiasi dato intercettato. TLS1.3 risolve questo problema disabilitando la crittografia di esportazione dei cifrari.

- *Factoring RSA Export Keys* (FREAK) è un attacco che consiste nell'ingannare il server per instaurare una connessione con una versione precedente di TLS (come SSLv2) che utilizza chiavi di cifratura deboli come quelle a 512 bit. TLS1.3 protegge da questo attacco disattivando il protocollo di downgrade, ovvero non permette di utilizzare una versione obsoleta di TLS.

4.2.2 Attacchi risultanti da un utilizzo di primitive crittografiche deboli

Molte vulnerabilità TLS riconosciute vengono generare da primitive crittografiche deboli, che vengono risolte da TLS1.3.

Alcuni di questi attacchi che sfruttano queste primitive crittografiche sono:

- *SWEET32* è un attacco contro CBC che utilizza 64 bit e sfrutta le collisioni dei blocchi. Utilizza una combinazione di attacchi compleanno e man in the middle o Javascript Injection contro un sito per facilitare l'instaurazione di un gran numero di richieste HyperText Transfer Protocol (HTTP). Questo può compromettere dati critici che possono essere mandati più volte, per esempio i token di autenticazione. Il blocco di cifrari Triple-DES (3DES) risolve queste problematiche.

- *Return of Bleichenbacher's Oracle Attack* (ROBOT) sfrutta la modalità di padding non sicuro come RSA-PKCS#1 v1.5 che consente di creare firme per certificati falsi. TLS1.3 risolve questo problema non

permettendo l'utilizzo di meccanismi di scambio di chiavi insicure come RSA-PKCS#1 v1.15

- *LUCKY13* è un attacco di timing crittografico contro l'implementazione di TLS fino alla versione 1.2 quando utilizza la modalità CBC. TLS1.3 risolve questa vulnerabilità non permettendo CBC.

4.2.3 Vulnerabilità di implementazione di TLS

Le vulnerabilità di implementazione del protocollo TLS sono numerose e alcune di queste danno origine ad attacchi che sfruttano un canale laterale.

Alcuni attacchi che sfruttano queste vulnerabilità sono:

- *Browser Exploit Against SSL/TLS* (BEAST) è uno dei principali attacchi contro TLS1.0. Sfrutta un utilizzo vulnerabile della modalità CBC (Cipher block Chaining) e degli Initialization Vector (IV). Permette all'attaccante, dato un numero di tentativi, di ottenere credenziali di autenticazione. In particolare, può ottenere token di sessione o i cookies di HTTP. L'utilizzo di TLS1.3 risolve il problema, non permettendo CBC.

- *Compression Ratio Info-leak Made Easy* (CRIME) è un protocollo che consiste in un attacco a canale laterale contro la compressione HTTPS. Questo permette di ottenere informazioni sulla compressione dei messaggi scambiati durante la comunicazione. TLS1.3 risolve il problema poiché non supporta più la compressione.

- *Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext* (BREACH) sfrutta il livello di compressione di HTTP per leggere il segreto di sessione dell'utente dal corpo del pacchetto HTTP. Funziona fino alla versione di TLS1.2 e per risolverlo occorre disattivare la compressione.

- *HTTP Encrypted Information can be stolen through TCP-Windows* (HEIST) combina gli attacchi contro TLS usando compressioni HTTP con un timing per canali laterali, utilizzando Javascript.

- *Security Losses form Obsolete and truncated transcript hashes* (SLOTH) è un attacco che permette di sfruttare le collisioni in funzioni hash vulnerabili. Per risolvere questo problema è necessario utilizzare funzioni hash più potenti come SHA256 ed evitare MD5 e SHA1.

- *Decrypting RSA with Obsolete and Weakened eNcryption* (DROWN) è efficace contro un server che supporti connessioni con un protocollo obsoleto come SSLv2. L'attaccante esegue questo attacco mandando false prove al server per convincerlo ad avviare una comunicazione SSLv2 e in questo modo, può per effettuare l'attacco.

- *Return of Coppersmith Attack* (ROCA) [13] permette ad un attaccante di recuperare la chiave privata dalla una chiave pubblica generata in modo non corretto, ovvero chiavi non scelte in modo casuale. L'attaccante, sapendo che la chiave non è generata casualmente ma è una tra le più utilizzate potrà utilizzare quest'informazione a suo favore. Essendo a conoscenza di queste informazioni, impiegherebbe meno tempo a calcolare la chiave privata. La scelta di non utilizzare chiavi pubbliche casuali è un errore comune e ha permesso una vasta diffusione di questo attacco.

4.2 Ricerca degli attacchi

4.2.1 Sniffing dei pacchetti

Lo sniffing è un'attività di intercettazione passiva dei dati che transitano nella rete. Questo tipo di attacco è utilizzato principalmente per l'analisi dei pacchetti scambiati tra diversi client e server. I principali software utilizzati per lo sniffing sono Wireshark [14] e TCPdump [15].

4.2.2 Address Resolution Protocol (ARP) poisoning

L'ARP poisoning [16] consiste nell'inviare intenzionalmente risposte ARP contenenti dati falsificati. In questo modo, la tabella ARP di un host contiene dati inesatti e un attaccante potrebbe sfruttarli per fingersi un altro utente della rete. Questo attacco è la base per effettuare il man in the middle in una LAN con dispositivi che operano al livello di rete del modello ISO/OSI. Il software utilizzato per l'applicazione di questo attacco è Arpspoof [17].

4.2.3 Attacco Man in the middle passivo

Nell'attacco Man in the middle (si faccia riferimento alla Figura 6) l'attaccante si interpone nella comunicazione tra client e server in modo trasparente, senza che i due ne siano a conoscenza. Per la realizzazione di questo attacco è necessario che venga corrotta l'associazione tra l'indirizzo IP e MAC del client e del server, i quali credono di comunicare tra loro con riservatezza. L'attaccante, una volta immesso con successo nella comunicazione, può ispezionare il traffico ed analizzarlo. L'attacco per definizione è detto passivo poiché i dati vengono solamente letti e analizzati ma non alterati. I software utilizzati sono Arpspoof con TCPdump, SSLsplit [18] ed Ettercap [19].

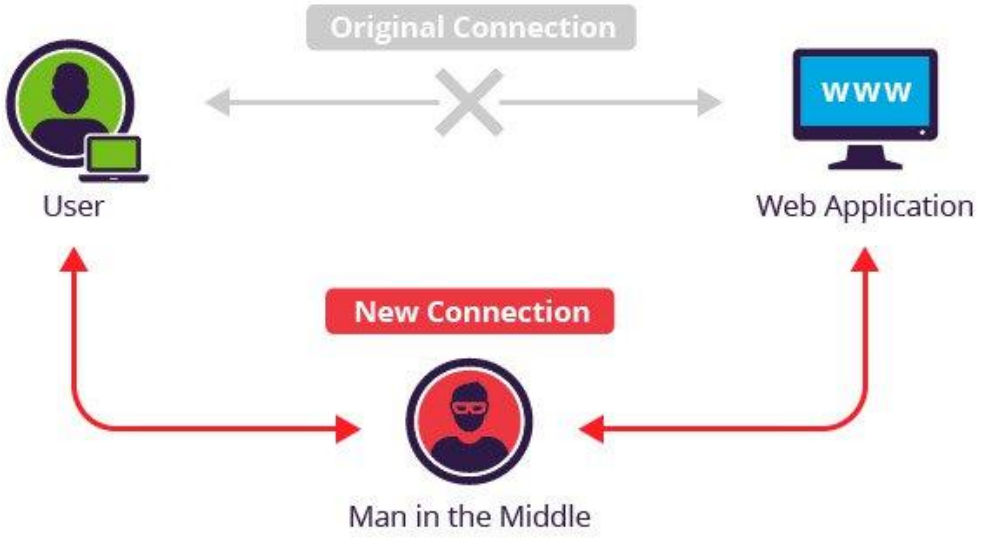

Figura 6: Illustrazione del man in the middle in pratica [20]

4.2.4 Attacco Websocket Denial of Service

L'attacco Denial of Service [21] consiste nell'instaurazione di una moltitudine di connessioni pendenti e concorrenti con un websocket server (si veda Figura 7), in modo da renderlo inutilizzabile. L'host infetto satura tutte le risorse disponibili e impedisce ai client veritieri di connettersi. Sono state definite due soluzioni possibili: uno script [22] realizzato con una la libreria di Python chiamata Scapy [23] e Websocket-bench [24].

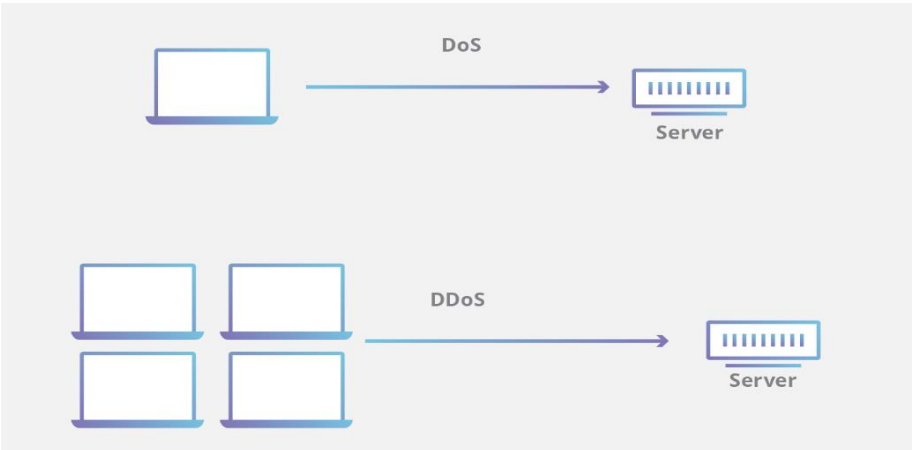

Figura 7: Schema di un attacco DoS e DDOS [25]

4.2.5 Attacco Downgrade

L'attacco downgrade è un attacco crittografico contro un sistema che utilizzi un certo protocollo di comunicazione in modo da obbligarlo a rifiutare una modalità qualitativamente migliore in favore di una obsoleta ed inferiore. Per questo tipo di attacco è stato utilizzato uno script [26] Python che utilizza la libreria NetfilterQueue [27] ma non è andato a buon fine.

4.2.6 Attacco Packet Filtering

L'attacco Packet Filtering consiste nell'instaurare un attacco man in the middle in modo tale da redirezionare il traffico nella macchina dell'attaccante, per poi rifiutare l'inoltro di determinati pacchetti (si veda Figura 8). L'attaccante può fare in modo di scatenare errori e problemi di comunicazione bloccando l'invio determinati comandi. I tool utilizzati per l'implementazione di questo tipo di attacco sono script in Python con l'utilizzo della libreria NetfilterQueue [27], che ha avuto un riscontro parzialmente positivo ed Ettercap, di cui però non è stata approfondita la funzione di filtering.

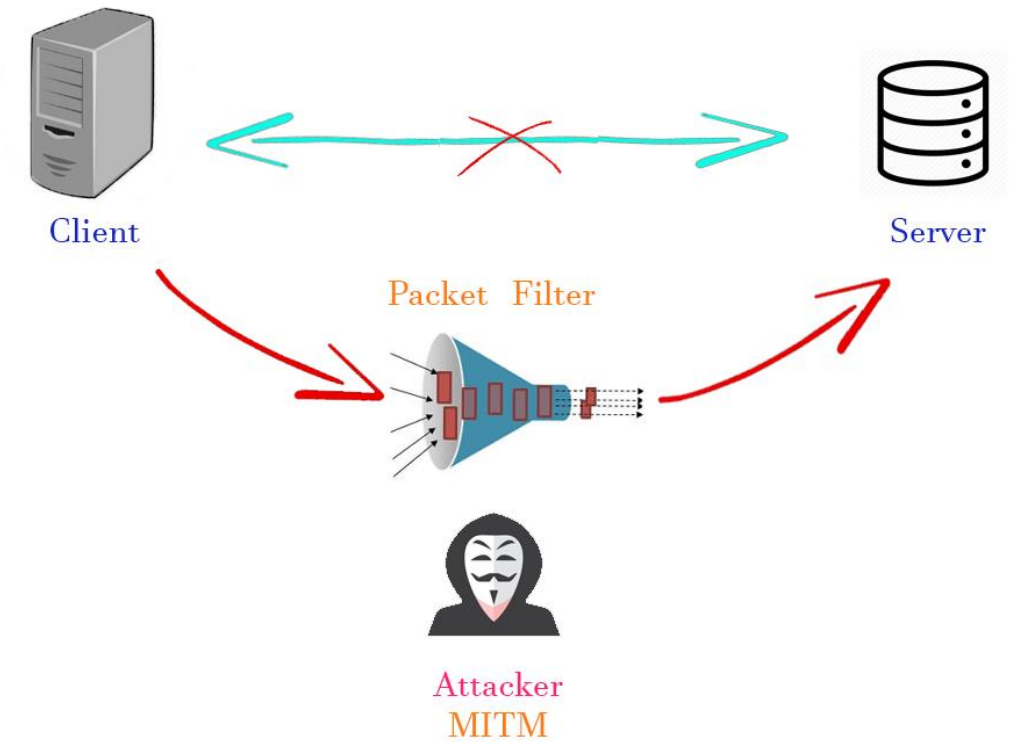

Figura 8: Funzionamento dell'attacco Packet Filtering

4.3 Strumenti di attacco

4.3.1 Strumenti per lo sniffing

Per la realizzazione dello sniffing di rete sono stati utilizzati:

- **Wireshark:** è un software di analisi di protocolli o Packet sniffer utilizzato per la risoluzione di problemi di rete. E' possibile analizzare i dati acquisiti in tempo reale su una rete ed è possibile salvarli su un file in formato Packet Capture (PCAP). Inoltre, supporta la codifica di un gran numero di protocolli tra cui MMS.

- **TCPdump:** è un tool comune per il debug delle reti che permette di intercettare pacchetti e trasmissioni di device connessi alla stessa rete. Per ottenere questo risultato è necessario che l'attaccante prenda il controllo di un router o di un gateway nel quale passi il traffico dell'intera rete.

4.3.2 Strumenti per l'ARP poisoning

Per la realizzazione di questo attacco è stato principalmente utilizzato:

- **ARPspoof**: è un tool che viene utilizzato per l'ARP poisoning e appartiene alla libreria dsniff.

4.3.3 Strumenti per l'attacco man in the middle passivo

Per la realizzazione del man in the middle passivo vi sono varie possibilità:

- **Ettercap** [19]: è un software che permette di intercettare i pacchetti, filtrare i contenuti in tempo reale e applicare l'attacco man in the middle. Supporta un grande numero di protocolli e può essere installato nella maggior parte dei sistemi operativi.
- **ARPspoof e TCPdump**: sono due software di attacco che, se utilizzati assieme, possono dar luogo ad un attacco man in the middle. ARP poisoning permette all'attaccante di immettersi nella comunicazione tra i due host e TCPdump, invece svolge la funzione di sniffer dei dati ricevuti dalle vittime.

- **SSLsplit** [18]: è un tool per l'applicazione di attacchi man in the middle contro connessioni che utilizzano la cifratura SSL e TLS. E' utile per network forensics, analisi di sicurezza per le applicazioni e penetration testing. Inoltre, per il protocollo HTTP e HTTPS è in grado di generare certificati on the go, utilizzabili per immettersi nella comunicazione tra due host.

4.3.4 Strumenti per Websocket DoS

Per la realizzazione di questo attacco sono stati utilizzati principalmente due tool:

- **Script Scapy** : Scapy è una libreria Python che permette di utilizzare funzioni per manipolare, decodificare o forgiare pacchetti di rete. Per l'applicazione dell'attacco DoS è stata utilizzata una funzione che permette di forgiare pacchetti SYN multipli per aprire connessioni TCP.

- **Websocket-bench**: è un tool open-source principalmente utilizzato per il testing di websocket server. Per questo tipo di attacco è stata sfruttata la possibilità di instaurare delle connessioni pendenti e concorrenti contro un server.

4.3.5 Strumenti per l'attacco Downgrade

Per la realizzazione di questo attacco è necessario l'utilizzo del tool:

- **NetfilterQueue:** è una libreria di Python che consente di accettare, rifiutare, manipolare e marchiare i pacchetti di rete. Per questo tipo di attacco, è stata utilizzata una funzione che consente di alterare il messaggio di client hello.

4.3.6 Strumenti per il Packet Filtering

Per l'applicazione del filtraggio dei pacchetti è stato utilizzato:

- **NetfilterQueue:** Per questo tipo di attacco è stata utilizzata la funzione di libreria che permette il blocco di determinati pacchetti inviati dal client.

4.4 Realizzazione degli attacchi

4.4.1 Realizzazione dello sniffing di rete

Lo sniffing di rete è stato realizzato con vari tool e hanno avuto tutti successo.

Tramite *Wireshark* è possibile intercettare e visualizzare tutti i pacchetti che il client e server MMS si sono scambiati durante la comunicazione. In particolare, è possibile eseguire un'analisi approfondita dell'handshake TLS tra i due host. Per fare questo è necessario specificare a Wireshark l'utilizzo di una decodifica dei pacchetti SSL che per default è impostata a TCP. Una volta impostata la decodifica specifica per SSL, è possibile visualizzare i dettagli dei messaggi di client e server hello. Da questi è possibile ottenere informazioni sui principali cipher suite e versioni del protocollo TLS supportati. E' emerso che la versione TLS supportata dal client e server MMS è la 1.1 con supporto ai seguenti cipher suites:

- RSA con AES256, CBC e SHA
- RSA con AES128, CBC e SHA

Il server MMS cercherà di instaurare una connessione con il client con l'utilizzo del primo cipher suite, il quale è qualitativamente superiore. Questo poiché viene utilizzato l'algoritmo di cifratura Advanced Encryption Standard (AES) con blocchi da 256 bit, migliore di quello con blocchi da 128 bit. Un attaccante con queste informazioni potrebbe eseguire una ricerca più approfondita delle vulnerabilità generate dalla scelta dei cipher suite e dalla scelta della versione di TLS da parte di client e server. Un'altra funzione di Wireshark è la possibilità di importare la chiave privata del server per poter decifrare la comunicazione. In un contesto reale, un attaccante esperto potrebbe conoscere la chiave privata e potrebbe decifrare e visualizzare tutti i messaggi in chiaro.

Con l'utilizzo di *TCPdump* è possibile intercettare i pacchetti, decodificarli e visualizzarli. A differenza di wireshark non possiede un'interfaccia grafica e un processo di analisi di rete potrebbe non essere intuitivo come con Wireshark. Dall'altro lato, un'interfaccia a livello di linea di comando permette di automatizzare task di avvio e di salvataggio dei dati intercettati.

4.4.2 Realizzazione dell'attacco ARP

L'attacco ARP è stato realizzato con successo tramite il tool *ARPspoof*, che ha permesso di alterare le tabelle ARP delle vittime. Quando il client tenta di connettersi al server, in realtà sta comunicando con l'attaccante, che successivamente inoltra i pacchetti ricevuti al destinatario di partenza. Per attivare l'inoltro è necessario impostare l'IP forwarding a 1 nella macchina dell'attaccante. In uno scenario di test in cui si ha la possibilità di accedere alle macchine delle vittime, è possibile verificare che l'attacco sia avvenuto con successo: è necessario visualizzare le tavole ARP dei due host infetti e verificare che ad uno stesso indirizzo Media Access Control (MAC) risultino assegnati due indirizzi IP differenti. Questo significa che quando un host infetto vuole definire una comunicazione con un altro host infetto, controlla le tabelle ARP (corrotte) e sceglie l'indirizzo MAC dell'attaccante come destinatario. Una delle principali criticità di questo attacco è una documentazione non dettagliata e precisa. Per esempio, in alcuni siti può essere specificato come target dell'attacco l'IP del gateway, mentre in altri casi l'indirizzo IP dell'attaccante. Infatti, in fase di test non è stato facile capire se fosse necessario specificare l'indirizzo del gateway o quello dell'attaccante come indirizzo di inoltro dei pacchetti intercettati. Questo perché alcune guide tenevano conto di un Man in the Middle nel quale il gateway si posizionasse nel mezzo della comunicazione ma non è questo il caso. Infatti, il container dell'attaccante risulta semplicemente un host della rete che finge l'host destinatario.

4.4.3 Realizzazione dell'attacco man in the middle

L'attacco man in the middle è stato realizzato con l'utilizzo di *ARPspoof e TCPdump*. Grazie al primo è possibile applicare l'ARP poisoning in modo che l'attaccante possa interporsi nella comunicazione tra client e server MMS; con il secondo, è possibile implementare il sistema di sniffing e intercettazione dei pacchetti tra i due host. Durante i test i tool vengono avviati in due container Docker differenti e per facilità realizzativa, vengono connessi alla rete Docker con lo stesso indirizzo IP e MAC.
La principale difficoltà incontrata durante l'applicazione di questo attacco è stata l'avvio in contemporanea di questi due strumenti ognuno in un container differente ma in modo da risultare una macchina singola.

E' possibile realizzare un attacco man in the middle anche con *Ettercap* che effettua tutte le operazioni necessarie in modo automatico. L'unica problematica riscontrata è l'avvio dell'applicazione con interfaccia grafica in un container Docker, poiché può dare luogo a problemi e generare errori. Inoltre, è necessario concedere dei permessi specifici a Docker, in modo che il sistema operativo della macchina host che ospita Docker, possa avviare l'interfaccia grafica dell'applicazione richiesta.

SSLsplit non permette di eseguire un attacco man in the middle tra client e server MMS. Il motivo principale è che SSLsplit supporta la creazione di certificati falsi [28], utilizzabili sul momento per instaurare una connessione fittizia solo in applicazioni che utilizzano il protocollo HTTP. Per tutti gli altri protocolli può solo effettuare operazioni di sniffing, che non sono intuitive come su Wireshark.

4.4.4 Realizzazione dell'attacco Websocket Denial of Service

L'attacco è stato realizzato con successo tramite il tool *Websocket-bench*. Con questo tool è possibile instaurare una moltitudine di connessioni con il server MMS, che avviato in un container Docker, accetta queste richieste e rifiuta quelle di client veritieri. Nei test sono bastate 20.000 connessioni totali di cui 2000 concorrenti al secondo per rendere il server inaccessibile dai client.

E' possibile realizzare un attacco DoS utilizzando uno script Scapy. In particolare, è l'implementazione di un attacco SYN flood, che però non ha portato a nessun risultato soddisfacente dato che non è riuscito a saturare con successo le connessioni del server. Infatti, sono necessarie connessioni specifiche per i websocket server.

4.4.5 Realizzazione dell'attacco Downgrade

Per la realizzazione dell'attacco Downgrade è possibile utilizzare la libreria NetfilterQueue. L'obiettivo è quello di intercettare e alterare il messaggio di client hello in modo tale da far risultare al server un cipher suite di qualità inferiore. Questa soluzione non si è conclusa con successo poiché il server è in grado di comprendere che il payload è stato alterato e non accetta nessuna richiesta dal quel client. In questo modo, l'handshake TLS non viene concluso con successo.

4.4.6 Realizzazione dell'attacco Packet Filtering

E' possibile realizzate l'attacco Packet Filtering con l'utilizzo della libreria NetfilterQueue. Sono stati definiti due script di attacco con due risultati differenti:

- **Connessione pendente**: vengono filtrati tutti i pacchetti inviati dal client dal ventisettesimo in avanti. In questo modo è possibile far concludere l'handshake TLS tra client e server MMS per poi filtrare tutti i pacchetti successivi, in modo tale che la connessione rimanga pendente e attiva. Questo script non è sempre funzionante, in alcuni casi il client può decidere di terminare la connessione poiché non riceve nessun messaggio di risposta da parte del server.
- **Errore di scrittura di report**: vengono filtrati tutti i pacchetti superiori ai 1500 byte, ovvero tutti quelli che in maggior parte corrispondono ai comandi inviati dal client. Nei test è stato possibile verificare come il client mandi un messaggio di richiesta di scrittura di un report che l'attaccante non inoltrerà al server. Quest'ultimo infatti non manderà l'*Acknowledge* (ACK) di risposta generando un errore di reporting nel client. La principale criticità riscontrata durante l'applicazione di questo attacco è stata la ricerca e selezione dei pacchetti più critici inviati dal client. Lo scopo di questa indagine è stato il generare il più grande numero di problemi possibili durante la comunicazione.

5. Conclusioni e sviluppi futuri

La sicurezza informatica sta acquisendo sempre più importanza negli ultimi anni, infatti vi sono ancora parecchi device e sistemi vulnerabili. Non tutti i dispositivi supportano una versione aggiornata di TLS1.3 e si stima che la maggior parte dei dispositivi supporti versioni più obsolete (si veda Figura 9).

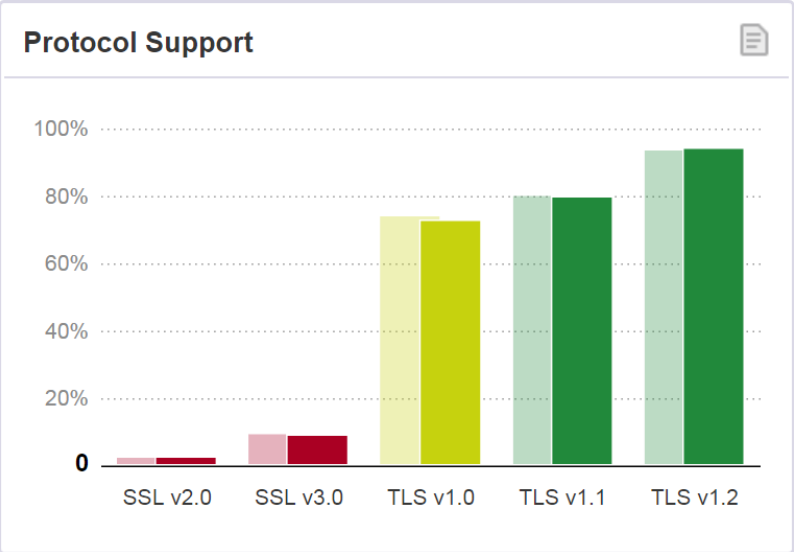

Figura 9: Supporto di TLS da parte dei sistemi [29]

Da un lato, negli uffici pubblici, ospedali e aziende non si tende a considerare questo aspetto come importante, dall'altro, gli attacchi informatici che avvengono quotidianamente sono in aumento (si veda Figura 10).

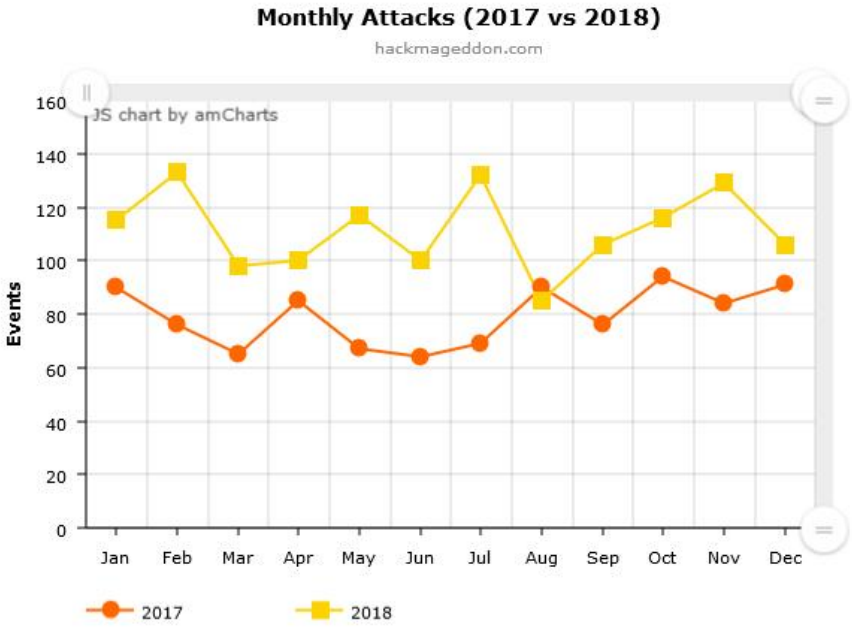

Figura 10: Attacchi mensili 2017 vs. 2018 [30]

In questa relazione sono state trattate proprio queste dinamiche e sono stati presentati vari attacchi contro il protocollo MMS con supporto a TLS1.1. Quelli implementati con successo sono stati: attacco man in the middle, DoS e Packet filtering. L'attacco Downgrade non è riuscito per problemi di accettazione dei pacchetti alterati da parte del server e per problemi di tempo non si è potuto approfondire la ricerca di altre soluzioni. Inoltre, non è stato possibile adattare SSLsplit al protocollo MMS per problemi di tempo e anche per complessità di struttura del codice.

Sebbene TLS1.1 sia nato nel 2006 e abbia ben 13 anni, molti dei dispositivi lo supportano e continuano ad utilizzarlo come prima scelta. Questi sistemi non sono considerati sicuri e anche TLS1.3 non può definirsi tale, poiché sono state trovate delle vulnerabilità. Nessun dispositivo è perfettamente sicuro.

6. Bibliografia

[1] «Ricerca sul Sistema Energetico,» [Online]. Available: http://www2.rse-web.it/home.page. [Consultato il giorno Agosto 2019].

[2] «Microsoft,» [Online]. Available: https://docs.microsoft.com/it-it/dotnet/architecture/microservices/container-docker-introduction/docker-defined.

[3] «Docker,» [Online]. Available: https://docs.docker.com/network/.

[4] «Docker,» [Online]. Available: https://docs.docker.com/network/bridge/.

[5] «Docker Docs,» [Online]. Available: https://docs.docker.com/v17.09/engine/userguide/networking/work-with-networks/#create-networks.

[6] N. Madiraju, «Nxgcloud,» [Online]. Available: http://nxgcloud.com/wp-content/uploads/2018/05/null-59.png.

[7] «Microsoft Docs,» [Online]. Available: https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc781476(v=ws.10).

[8] «IBM,» [Online]. Available: https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_7.1.0/com.ibm.mq.doc/sy10660_.htm.

[9] E. C. CCE-CNMA, MMS: A Communication Language for Manufactoring, Springer.

[10] «Sisco,» [Online]. Available: http://www.sisconet.com/wp-content/uploads/2016/03/mmsovrlg.pdf.

[11] «Libiec61850 Docs,» [Online]. Available: https://libiec61850.com/libiec61850/documentation/.

[12] «Cloudinsidr,» [Online]. Available: https://www.cloudinsidr.com/content/known-attack-vectors-against-tls-implementation-vulnerabilities/.

[13] A. Beaupré, «LWN.net,» [Online]. Available: https://lwn.net/Articles/738896/.

[14] «Wireshark Docs,» [Online]. Available: https://www.wireshark.org/docs/.

[15] «Manage of TCPdump,» [Online]. Available: https://www.tcpdump.org/manpages/tcpdump.1.html.

[16] «Technopedia,» [Online]. Available: https://www.techopedia.com/definition/27471/address-resolution-protocol-poisoning-arp-poisoning.

[17] S. Uhlmann, «su2,» [Online]. Available: https://su2.info/doc/arpspoof.php.

[18] «Roe,» [Online]. Available: https://www.roe.ch/SSLsplit.

[19] «Ettercap Project,» [Online]. Available: https://www.ettercap-project.org/.

[20] «Imperva,» [Online]. Available: https://www.imperva.com/learn/application-security/man-in-the-middle-attack-mitm/.

[21] «Cybersecurity360,» [Online]. Available: https://www.cybersecurity360.it/nuove-minacce/ddos-cosa-sono-questi-attacchi-hacker-e-come-stanno-evolvendo/.

[22] S. V. Pachghare, «Opensourceforu,» [Online]. Available: https://opensourceforu.com/2011/10/syn-flooding-using-scapy-and-prevention-using-iptables/.

[23] «Scapy,» [Online]. Available: https://scapy.net/.

[24] sosоsoyoung, «GitHub,» [Online]. Available: https://github.com/sosоsoyoung/websocket-bench.

[25] «Cloudflare,» [Online]. Available: https://www.cloudflare.com/learning/ddos/glossary/denial-of-service/.

[26] L. Barman, «LBarman,» [Online]. Available: https://lbarman.ch/blog/downgrade-tls/.

[27] «Pypi,» [Online]. Available: https://pypi.org/project/NetfilterQueue/.

[28] P. C. Heckel, «Philipp's Tech Blog,» [Online]. Available: https://blog.heckel.io/2013/08/04/use-sslsplit-to-transparently-sniff-tls-ssl-connections/.

[29] L. Abrams, «Bleepingcomputer,» [Online]. Available: https://www.bleepingcomputer.com/news/security/tls-10-and-tls-11-being-retired-in-2020-by-all-major-browsers/.

[30] P. Passeri, «Hackermageddon,» [Online]. Available: https://www.hackmageddon.com/2019/01/15/2018-a-year-of-cyber-attacks/.

7. Ringraziamenti

A conclusione di questa relazione, è doveroso ringraziare le persone che ho avuto modo di conoscere in questi tre anni e che mi hanno aiutato a crescere sotto ogni aspetto.

Vorrei cominciare col ringraziare la Prof.ssa Lavinia Egidi, tutore interno durante il mio studio. Oltre ad avermi guidato nella stesura di questa relazione, ha saputo orientarmi durante lo svolgimento del mio tirocinio con estrema disponibilità. Inoltre, ringrazio l'azienda RSE per avermi dato l'opportunità di svolgere l'attività di tirocinio. Un ringraziamento particolare a mia madre e mio padre che, grazie al loro sostegno sia morale che economico, mi hanno permesso di poter raggiungere questo traguardo. In particolare a mia madre, che mi ha sempre supportato nella realizzazione dei progetti accademici con il seguente consiglio: "prova a spegnere e riaccendere". Un grazie enorme anche a mio fratello, ai miei amici e ai miei colleghi per il sostegno e l'incoraggiamento dato. Un pensiero speciale anche a Ginevra che, grazie al suo affetto e sostegno, ha reso questo traguardo ancora più prezioso e, nonostante le paure, ha saputo incoraggiarmi più di chiunque altro.

Un grazie di cuore a tutti!

Andrea Ierardi