ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

SCUOLA DI SCIENZE
Corso di Laurea in Informatica

Analisi delle Tecniche di Cifratura nei Ransomware rispetto alla Disponibilità di Memoria Secondaria

Relatore:
Chiar.mo Prof.
SAVERIO GIALLORENZO

Correlatore:
Dott.
SIMONE MELLONI

Presentata da:
BERNARDO MACCHIONI MONTINI

II Sessione
Anno Accademico 2023/2024

Abstract

L'elaborato affronta le tematiche legate ai ransomware, categoria di malware che cripta i file sul disco chiedendo poi un pagamento per potervi riavere accesso. L'obiettivo è analizzare vari campioni di questi virus per comprenderne meglio il comportamento nella situazione in cui lo spazio nella memoria secondaria sia esaurito, in modo da aiutare futuri sviluppi nei relativi strumenti di contrasto.

La prima sezione prevede un'introduzione agli strumenti utilizzati nel corso dell'analisi per automatizzare e rendere più efficienti le operazioni, ovvero *Proxmox*, *Terraform*, *Ansible* e *Filechecker*.
La sezione successiva riguarda la prima parte del lavoro, che consiste nella ricerca di materiale sul tema, ad esempio, per capire quali esemplari di ransomware analizzare e trovarne dei campioni.
In seguito, viene spiegato il modus operandi: la preparazione delle macchine virtuali, l'esecuzione dei test e l'analisi dei risultati.
Infine vengono discussi i test, come sono strutturati e come sono eseguiti, e si analizzano e interpretano i risultati ottenuti.

Indice

Capitolo 1

Introduzione

L'avvento della digitalizzazione e, in particolare, lo sviluppo esponenziale dell'informatica e di Internet, hanno portato innumerevoli benefici alla società moderna. La possibilità di accedere in tempo reale a informazioni, comunicare istantaneamente con persone in ogni angolo del pianeta e gestire una vasta gamma di operazioni quotidiane attraverso dispositivi elettronici, ha rivoluzionato sia il modo di vivere che quello di lavorare.
Oggi, l'utilizzo di computer, smartphone e altri dispositivi digitali è ampiamente diffuso, non solo a livello personale ma anche in ambito aziendale e istituzionale. Tuttavia, questa crescente dipendenza dai supporti digitali ha anche creato nuove opportunità per attori malevoli.

Con la sempre maggiore diffusione dei dispositivi informatici e della connettività, sono infatti emerse numerose minacce alla sicurezza informatica. Tra queste figurano i malware, "malicious software", ovvero dei programmi progettati per arrecare danni agli utenti in vari modi, ad esempio rubando dati sensibili, compromettendo l'integrità dei sistemi o impedendo l'accesso a servizi cruciali, come nel caso degli attacchi (D)DoS, (Distributed) Denial of Service.

Un tipo specifico di malware che ha guadagnato particolare notorietà negli ultimi anni è il ransomware.
I ransomware attaccano i sistemi informatici tipicamente criptando i file presenti nella memoria del dispositivo della vittima, rendendoli inaccessibili. Gli aggressori quindi chiedono un riscatto in cambio della chiave di decrittazione, lasciando la vittima di fronte alla difficile scelta fra pagare o perdere definitivamente i propri dati. Questo tipo di malware è particolarmente distruttivo, poiché non si limita a compromettere il sistema, ma colpisce direttamente i dati personali o aziendali, causando danni potenzialmente irreparabili.

Affinché un ransomware possa eseguire con successo la crittografia dei file, è necessario che effettui delle operazioni sul disco, sostituendo i file originali dell'utente con le controparti criptate.
Tuttavia, sorgono dei dubbi su quale sia il comportamento di un ransomware quando il disco sotto attacco è pieno. Questo aspetto apre una serie di interrogativi: il malware si arresta e non esegue ulteriori operazioni? Oppure potrebbe sovrascrivere direttamente i file esistenti, proseguendo nella sua azione distruttiva?

Lo scopo di questo studio è proprio quello di indagare il comportamento dei ransomware in situazioni di memoria piena. Cercheremo di comprendere come questi malware reagiscano a limitazioni di spazio sul disco, il che potrebbe fornire indicazioni utili per migliorare le strategie di difesa contro questo tipo di minacce, offrendo nuovi spunti per la prevenzione e la mitigazione degli attacchi ransomware.

Per affrontare questa indagine in maniera rigorosa, è necessario raccogliere un volume consistente di dati, poiché il comportamento dei ransomware potrebbe variare in base a diversi fattori, come la configurazione del sistema, il tipo di file presenti o le specifiche del malware stesso. Per ottenere risultati statisticamente significativi, è dunque essenziale condurre un numero di prove considerevole su ambienti controllati e replicabili, simulando scenari in cui i dischi siano completamente pieni o in prossimità della saturazione.

Tuttavia, eseguire manualmente un numero così elevato di test sarebbe estremamente oneroso in termini di tempo e risorse. Per questo motivo, si rende necessario l'utilizzo di una piattaforma automatizzata in grado di gestire un gran numero di esecuzioni.
In questa tesi viene utilizzato e raffinato tale sistema per eseguire test in modo rapido ed efficiente e per ridurre al minimo gli errori umani, garantendo la ripetibilità delle condizioni e una maggiore affidabilità dei risultati.

Capitolo 2

Tecnologie e strumenti utilizzati

In questo capitolo procederemo ad introdurre le tecnologie e i concetti che sono serviti durante il lavoro di tesi e necessari per la sua comprensione, e i relativi strumenti usati per implementarli.

2.1 Virtualizzazione

La virtualizzazione, nel campo informatico, è un metodo utilizzato per fornire delle risorse virtuali al software, astraendo le componenti hardware del computer.

Questo metodo, tramite l'uso del software, crea un livello di astrazione rispetto all'hardware, le effettive componenti fisiche dell'elaboratore, come processore e memoria primaria e secondaria.
Questo concede la possibilità di creare delle "macchine virtuali" o "virtual machine" (VM), ovvero dei computer virtuali, ed eseguire su ognuna di queste un Sistema Operativo a scelta.

Tutte queste macchine virtuali, in realtà condividono e sono in esecuzione sulle uniche componenti hardware reali, quelle del computer su cui sono installate, ma funzionano come se avessero il proprio hardware grazie alla virtualizzazione e all'astrazione del livello fisico.

Per fornire queste funzionalità si fa affidamento su un *hypervisor*, l'elemento software che gestisce le macchine virtuali e le loro interazioni con l'hardware.
Gli hypervisor possono essere di due tipi, detti tipo 1 e 2. Il primo è installato direttamente sull'hardware, mentre il secondo si trova in esecuzione su un sistema operativo, come un normale programma.

Ad esempio, rientrano nel tipo 1 gli hypervisor *Microsoft Hyper-V* e *Proxmox*, mentre *VirtualBox* e *VMware Workstation* sono della seconda tipologia.

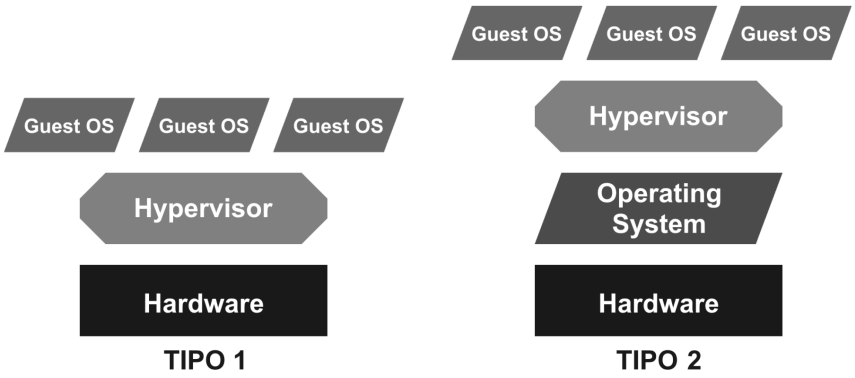


Figura 2.1: Tipi di hypervisor

La virtualizzazione ci viene in aiuto col nostro problema. In particolare, permette di eseguire prove sui ransomware su macchine virtuali, dove non sono presenti dati sensibili o informazioni importanti per l'utente, dato che sono state appositamente create per questo studio; inoltre isolano il sistema operativo e i loro processi interni, il che consente di controllare la diffusione del malware.

2.1.1 Proxmox

Per gestire l'insieme delle macchine virtuali è stato impiegato *Proxmox*, che fa parte della prima tipologia di hypervisor. Si tratta di una distribuzione Linux basata su Debian che permette, tra le varie funzionalità, di creare e clonare macchine virtuali e template, ovvero modelli da usare come riferimento per generare altre macchine.
Proxmox ci consente di portare a termine i nostri studi in ambienti isolati in cui solo la risorsa virtuale può essere danneggiata.

2.2 Infrastracture as Code

Il termine infrastruttura, in informatica, si riferisce alla struttura di un sistema, cioè ai vari elementi, sia hardware che software, che lo compongono. Questi possono comprendere ad esempio computer, router, server e database.
In uno studio come quello qui presentato si ha la necessità di apportare frequentemente modifiche all'infrastruttura, il che rende quantomeno scomodo e dispendioso il provisioning, cioè il processo con cui si prepara l'infrastruttura e la si rende disponibile all'utilizzo.

Anche per questo motivo negli anni si è sviluppata la *Infrastructure as Code* che, come suggerisce il nome, consiste nell'automatizzare il più possibile, tramite il codice, il provisioning, cosicché si possa risparmiare tempo ed evitare eventuali errori umani dovuti alla procedura manuale.

2.2.1 Terraform

Terraform è lo strumento di Infrastructure as Code che è stato utilizzato e con il quale si possono facilmente creare nuove macchine virtuali specificandone in un file le caratteristiche desiderate.
Il funzionamento di Terraform si basa sui provisioner, plugin che comunicano col servizio in uso, nel nostro caso Proxmox, e le risorse, ovvero ciò che deve essere creato, come le macchine virtuali.

2.3 Automazione delle procedure

Dato l'elevato numero di prove da effettuare è indispensabile che queste vengano automatizzate, per un'esecuzione più efficiente.

OS-agnostic Task Automation è un termine che indica un tipo di automazione che è indipendente dal Sistema Operativo specifico su cui sarà impiegata, funzionando a un livello di astrazione superiore e concentrandosi sui task da eseguire.

2.3.1 Ansible

Ansible è uno strumento open-source di automazione IT utilizzato per provisioning, deployment di applicazioni, gestione della configurazione e molti altri processi IT.
Utilizzando i protocolli SSH o Windows Remote Management, rientra nella tipologia *agent-less*, perché non richiede l'installazione di software sulla macchina da gestire.

Si basa su due elementi, controller e host, dove il primo è la macchina da cui vengono diramati i comandi che vengono poi eseguiti sul secondo.
I file richiesti a questo scopo sono il playbook, in formato YAML, dove sono definiti i compiti da attuare, l'inventario, in cui si impostano gli indirizzi IP degli host, e opzionalmente un file aggiuntivo dove dichiarare delle variabili che possono essere riutilizzate nel playbook.

2.4 Strumenti di contrasto

A causa dell'aumento degli attacchi con ransomware negli ultimi anni, l'interesse per strumenti di contrasto specifici per questa tipologia di malware è cresciuto no-

tevolmente.
Uno dei software nati da questa esigenza è *Ranflood*.

2.4.1 Ranflood

Ranflood[BGM+24] è uno strumento per la mitigazione degli effetti dei ransomware e il ripristino della situazione precedente all'attacco.
L'idea su cui si basa è il *Data Flooding against Ransomware*[BGM+23], una tecnica che consiste nel creare dei file esca per contrastare il virus in due modi: contendere l'accesso al disco e far "sprecare" tempo al ransomware nella criptazione dei file generati.

Ranflood è composto da due unità: un demone, che deve essere attivato inizialmente e poi lasciato attivo, e il client, da cui si possono impartire i comandi, come lo start e lo stop del contrasto.

Sono state implementate più strategie:

- Random
  La più basilare, prevede solamente la generazione di file sul momento.
- On-The-Fly
  Espande la prima, rimpiazzando la generazione di file casuali con la copia di file dell'utente per aumentare la probabilità che almeno un'istanza del documento, sia essa l'originale o una copia, venga preservata.
- Shadow
  La più avanzata ma anche la più impattante sulle risorse, in quanto richiede il salvataggio di uno snapshot con i file da proteggere.

Per il nostro studio Ranflood sarà usato non per la sua capacità di contrastare i ransomware, bensì per riempire, tramite il flooding in modalità random, il disco delle macchine prima di eseguire i virus, cosicché da trovarsi nella condizione di memoria piena, a cui siamo interessati.

2.5 Strumenti per l'analisi

Dopo il completamento dei test sarà necessario analizzare i risultati.
Questa operazione sarà effettuata controllando quali file, tra quelli originariamente memorizzati sulla macchina, sono ancora presenti e non criptati.

2.5.1 Filechecker

Per questo scopo useremo *Filechecker* [BGM+23], uno strumento apposito, che permette di fare questa verifica.
Il funzionamento di Filechecker è fondato sui checksum, stringhe alfanumeriche calcolate da un algoritmo per ogni file e che permettono di identificarli.
Il suo utilizzo si divide in due passaggi: nel primo si procede a eseguire un comando per salvare i checksum dei file di cui si vuole tenere traccia, nel secondo, con un altro comando, si confrontano i file attuali con i checksum salvati nel primo passo per capire quanti sono andati persi a causa del malware.

Per ogni file si possono avere tre condizioni differenti:

- pristine, ovvero il file non è stato toccato dal ransomware
- replica, il file è stato replicato, ad esempio con una delle strategie che prevedono la copia, oppure dal virus, durante le sue operazioni preliminari in preparazione alla cifratura
- lost, il file è perso/criptato

Capitolo 3

Ricerca della letteratura

Prima di poter svolgere qualsiasi attività è necessario documentarsi sulla materia di cui ci si deve occupare, quindi come prima cosa è stata svolta una ricerca della letteratura sui ransomware e in particolare delle caratteristiche che possono influenzare il loro comportamento in caso di scarsa disponibilità di memoria secondaria.

3.1 Tassonomia dei ransomware

Inizialmente sono stati analizzati dei paper [OALU22] [GAF+15] [ArMS18] che propongono una tassonomia dei ransomware per poterli categorizzare; ognuno di questi si è basato su varie caratteristiche del malware, quali, ad esempio, la piattaforma attaccata, il metodo di infezione, se si tratta di crypto ransomware, i più diffusi e di cui ci occuperemo, o di locker ransomware, che non cripta i file ma blocca l'utilizzo del dispositivo.

Nonostante abbiano fornito varie informazioni sulle proprietà dei ransomware, questi articoli non si sono rivelati particolarmente utili, poiché non erano presenti dati che potessero indicare la modalità di esecuzione del virus in condizioni come quelle affrontate in questo studio.

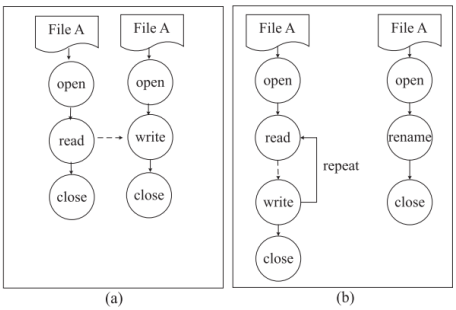
3.2 Pattern di accesso ai file

La necessità di comprendere il funzionamento dei ransomware quando posti di fronte ad una situazione di poco spazio disponibile su disco ha portato ad altri tre articoli [TML+20] [KAM+16] [WJC+19].

Questi riportano, in relazione ai campioni analizzati, i pattern di accesso ai file durante la loro attività, che è ciò di cui abbiamo bisogno.

3.2.1 RansomSpector

Il primo degli articoli[TML+20] propone una suddivisione in base a tre pattern di accesso ai file.
Siamo interessati ai primi due perché sono coinvolte operazioni che potenzialmente non hanno bisogno di ulteriore spazio sul disco, dato che i file sono direttamente sovrascritti senza crearne copie locali, e di conseguenza potrebbero continuare a criptare anche in assenza di disponibilità di memoria.



(a) Schema dei due pattern rilevanti

Statistics of ransomware behavior patterns.

Pattern	File I/O Access		
	a	*b*	*c*
number	24	709	38
rate	3.11%	91.96%	4.93%

(b) Distribuzione dei campioni in base ai pattern

Figura 3.1: I pattern di RansomSpector

Purtroppo non viene indicato a quale categoria appartengono i campioni analizzati ma è solamente fornita la loro distribuzione percentuale, da cui possiamo comunque notare che la maggior parte di questi ricadono nei pattern a e b.

3.2.2 UNVEIL

Un altro paper[KAM+16] fa una distinzione simile, ma in questo caso il pattern rilevante per noi è uno solo, dal momento che gli altri creano nuovi file per portare a termine la criptazione, occupando altro spazio.

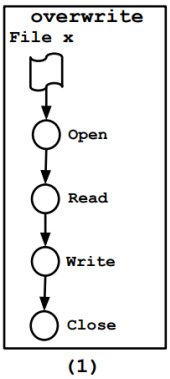


Figura 3.2: Schema del pattern

In questo caso non è riportata la distribuzione come nell'articolo precedente ma solo alcuni esempi di famiglie di ransomware in relazione al loro pattern.
CryptoLocker e alcuni campioni di CryptoWall fanno parte del tipo 1.

3.2.3 MimosaFTL

L'ultimo articolo consultato[WJC+19] individua quattro diversi pattern, fra i quali i tipi A e B applicano la sovrascrittura saltando il passaggio di copia dei dati. Vengono fornite le statistiche relative a tutti i campioni studiati, che rendono chiaro quali ransomware fanno parte delle categorie d'interesse.

Family	#Samples	A	B	C	D	Time
TeslaCrypt	26(5.02%)	145	5	0	0	12
Locky	131(25.29%)	137	1	9	3	12
Cerber	23(4.44%)	143	4	3	0	2
Ransom32	28(5.40%)	131	8	11	0	8
CTB-locker	71(13.71%)	2	141	5	2	2
CryptoLocker	49(9.46%)	1	146	2	1	2
HydraCrypt	38(7.34%)	0	1	121	28	12
Samas	30(5.79%)	0	0	31	119	19
Bart	6(1.16%)	2	4	10	134	2
CryptoWall	102(19.69%)	1	7	15	127	13
Maktub	14(2.70%)	0	3	15	132	4
Total	518	-	-	-	-	-

Figura 3.3: Distribuzione dei ransomware sui vari pattern

3.3 Rilevazione dei pattern

Approfondiamo ora, dopo aver discusso i risultati di queste ricerche, come sono stati ottenuti, ovvero in che modo sono stati osservati i pattern di accesso ai file dei ransomware.

3.3.1 RansomSpector

RansomSpector, come è stato chiamato dagli autori, è basato su un hypervisor sul quale viene virtualizzato il sistema operativo.
Come si può intuire, ciò permette di catturare le system call che vengono effettuate, salvare queste informazioni e successivamente elaborarle per estrapolarne la modalità di accesso ai dati.

3.3.2 UNVEIL

Anche il sistema adottato dai ricercatori del secondo paper, denominato *UNVEIL*, è concettualmente abbastanza semplice: usa infatti dei driver del kernel Windows modificati che garantiscono la possibilità di monitorare il filesystem.

3.3.3 MimosaFTL

Infine, l'ultimo articolo è quello che impiega l'approccio più complesso.
Non vengono direttamente controllati gli accessi ai file ma agli LBA (Logical Block Addresses).
Per fare questo, gli autori hanno collegato una scheda di sviluppo elettronica come memoria di massa USB al dispositivo su cui è installata la macchina virtuale e con uno strumento di debugging seriale hanno rilevato gli accessi.
Dopo di che, per categorizzare i campioni è stato fatto ricorso al metodo di learning non supervisionato K-means clustering.

3.4 Selezione dei ransomware

Completato lo studio della letteratura è stato necessario ricercare gli esemplari di ransomware su cui svolgere gli esperimenti.

I campioni sono stati selezionati dalle repository *theZoo* [yti], *Malware Database* [Gay] e dalla repository di *Ranflood* [BGM+24] [BGM+].

Purtroppo, fatta eccezione per gli esemplari riportati da MimosaFTL, non vengono esplicitamente menzionate quali famiglie di ransomware ricadono nei vari pattern;

tuttavia questo non è un problema considerato che secondo gli studi menzionati molti ransomware procedono con la criptazione dei file evitando l'utilizzo di ulteriore memoria.
Inoltre condurre le attività con campioni di entrambe le categorie può essere d'interesse ai fini dello studio per analizzarne le differenze.

3.4.1 Campioni non funzionanti

Sfortunatamente molti ransomware tendono a "nascondersi" e non compiere alcuna azione se individuano una situazione anomala che interptretano come segnale della possibilità di star venendo analizzati, questo per evitare di fornire informazioni sul proprio funzionamento ai ricercatori.
Conseguentemente si è reso indispensabile sperimentare con molti di essi per identificare un numero sufficiente di esemplari che non si bloccassero nel nostro ambiente di prova.

Le famiglie rilevate nei pattern A e B da MimosaFTL, ad esempio, non sono state utilizzate nello studio per il motivo sopracitato.

I campioni di ransomware funzionanti e analizzati in questo lavoro appartengono alle seguenti famiglie:

- *7ev3n*
- *Annabelle*
- *Birele*
- *Fantom*
- *WannaCry*
- *Winlocker*

Capitolo 4

Preparazione dei test

Nella preparazione del template, il provisioning con Terraform e l'utilizzo di Ansible sono state seguite le indicazioni della tesi sullo sviluppo di SAFARI [Com23], un framework per indagine automatizzata sui ransomware.

4.1 Cluster

Un cluster di computer è un insieme di computer connessi tra loro tramite una rete telematica. Lo scopo di un cluster è distribuire un'elaborazione molto complessa tra i vari computer, aumentando la potenza di calcolo del sistema e/o garantendo una maggiore disponibilità di servizio. [Wika]

Per realizzare le prove, era disponibile un cluster su cui è installato Proxmox, a cui ci riferiremo d'ora in poi semplicemente come "rig", col quale si potevano gestire le VM.

4.2 Preparazione delle macchine virtuali

4.2.1 Caratteristiche del template

I test sono stati eseguiti su delle macchine virtuali con sistema operativo Windows 10 installate sul rig con Proxmox.
Il progetto di SAFARI fornisce già un template Windows 10 popolato con un profilo utente con dei file "artificali" generati appositamente nelle cartelle utente (Desktop, Download, Immagini, ecc...) allo scopo di simularne l'utilizzo da parte di un utente medio, ma erano ancora liberi vari gigabyte di memoria secondaria, che è stata ulteriormente occupata con altri documenti trasferiti tramite SCP [Wikb].

In questo modo si ottiene una VM con memoria libera ridotta ma non totalmente piena, altrimenti non si potrebbero portare a termine gli esperimenti, per cui serve dello spazio, seppur poco.
Infatti, oltre ai già menzionati, sono stati copiati sulla macchina altri file, come gli eseguibili di Ranflood, per riempire completamente il disco prima di eseguire il ransomware, e uno script batch per invocare daemon e client di Ranflood.

4.2.2 Clonazione con Terraform

Dopo aver preparato la VM (Virtual Machine) con tutto il necessario, è stata convertita in template affinché potesse essere clonata ad ogni esigenza per creare nuove macchine su cui effettuare i test.

Come già spiegato, per la creazione di nuove VM è stato fatto affidamento su Terraform, con cui si può automatizzare e velocizzare il processo.

A questo scopo serve definire alcuni file:

- **provider.tf**
  In questo file si definiscono il provider, la versione richiesta di quest'ultimo e di Terraform e anche le credenziali per l'autenticazione da parte del provider presso Proxmox.
  Come provider è stato scelto Telmate, uno dei più diffusi per Proxmox.

```
terraform {
    required_version = ">= 1.6.1"
    required_providers {
        proxmox = {
            source = "telmate/proxmox"
            version = "2.9.14"
        }
    }
}

provider "proxmox" {
    pm_api_url = var.proxmox_api_url
    pm_api_token_id = var.proxmox_api_token_id
    pm_api_token_secret = var.proxmox_api_token_secret
}
```

- **windows_clone.tf**
  Qua si definiscono il numero di VM da creare e i loro parametri, ad esempio il nome e l'ID, le caratteristiche hardware e il template da clonare.

```
resource "proxmox_vm_qemu" "windows_clone" {
    count = var.vms-count

    name = "${var.vm_name}-${count.index}"
    target_node = var.target_node
    vmid = sum([var.vm_id, count.index])
    clone = var.template_clone
    bios = var.vm_bios
    ipconfig0 = var.vm_ip
    full_clone = true

    sockets = var.vm_sockets
    cores = var.vm_cores
    memory = var.vm_memory

    scsihw = var.vm_scsi

    disk {
        type = var.vm_disk_type
        storage = var.vm_disk_storage
        size = var.vm_disk_size
    }

    network {
        model = var.network_card_model
        bridge = var.network_bridge
    }
}
```

- **terraform.auto.tfvars**
  Opzionalmente, come fatto in questo caso, si può creare un altro file in cui dichiarare le variabili da usare nei precedenti.

4.2.3 Esecuzione dei test con Ansible

Per i test è stato fatto ricorso ad Ansible e si è configurato il template per funzionare sia come host Ansible che come controller, installando Ubuntu con WSL [Mic].
Questo perché Ansible può essere installato solo su macchine con sistema operativo Unix-like.

Il bisogno di far funzionare la VM sia come host che come controller Ansible deriva dalla logica dei test con Ansible.
In primo luogo viene eseguito un playbook dalla propria macchina, il nodo controllore, che serve a trasferire i file necessari, tra cui un playbook "interno" che viene

lanciato da quello "originario" in maniera asincrona, sulla VM su cui eseguire le prove, che funziona allo stesso tempo da host e da controller.
Per l'ultimo passo descritto serve che sulla macchina virtuale sia presente WSL, su cui si può installare Ansible, in modo da eseguire il playbook.

L'approccio descritto serve per sfruttare il concetto di *Air Gap*, una misura di sicurezza basata su isolamento logico e/o fisico, che è stato usato per separare le VM ed evitare la diffusione dei ransomware.

Riempimento del disco con file batch

Come già accennato, per avere la memoria secondaria completamente occupata, il playbook interno, prima di eseguire il ransomware, lancia uno script batch che avvia Ranflood; questo procede ad effettuare il flooding nella cartella specificata e a impiegare lo spazio che era ancora rimasto libero.

Capitolo 5

Caso di studio

Durante lo studio sono stati analizzati vari ransomware.
In primo luogo, sono stati selezionati gli esemplari indicati nell'ultimo articolo consultato [WJC+19], poi scartati dopo una prima verifica, con esito negativo, per accertarsi che si attivassero.
In seguito, la stessa procedura è stata applicata ad altri campioni provenienti da varie repository [yti] [Gay] [BGM+].

Infine, sono stati scelti 6 ransomware, tutti ottenuti da Malware Database [Gay], ovvero 7ev3n, Annabelle, Birele, Fantom, WannaCry e Winlocker.

Il caso di studio si pone come obiettivo l'analisi del comportamento dei ransomware in mancanza di disponibilità di spazio su disco.

Per ogni ransomware sono state eseguite quattro prove e in ognuna di queste la macchina virtuale è stata lasciata accesa per 8 minuti dopo l'avvio del playbook interno. È stato impostato questo intervallo dopo aver verificato che fosse sufficiente affinché il virus potesse eseguire la criptazione.

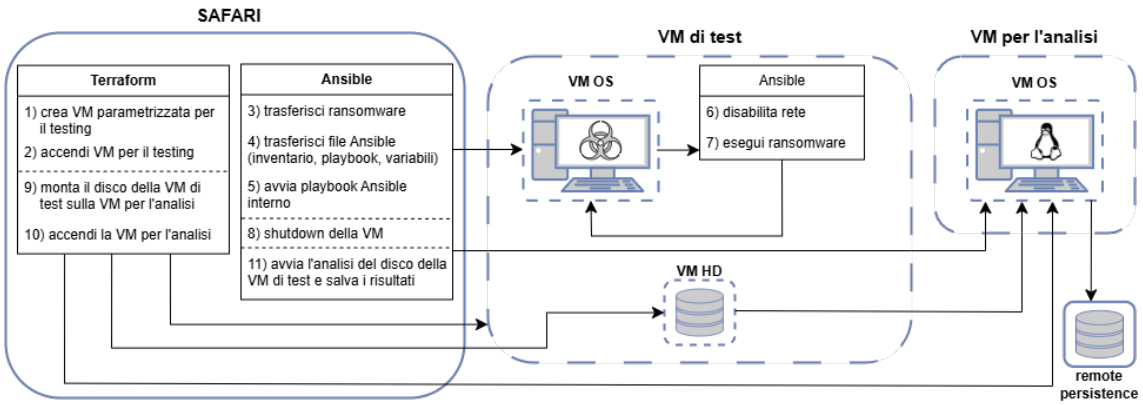


Figura 5.1: Architettura del sistema

5.1 Organizzazione di Ansible

Per condurre i test è stato usato Ansible e sono stati definiti quattro file principali:

- **full_playbook.yml**
  Questo è il playbook che viene eseguito dalla propria macchina.
  Si collega alla VM e trasferisce i file di Ansible e il ransomware, dopo di che fa partire il playbook interno in modo asincrono e, dopo un timeout, spenge la macchina.

```
1  - hosts: windows
2    tasks:
3
4      - name: Trasferisci ransomware compresso
5        ansible.windows.win_copy:
6          src: "{{ rf_fulldisk_dir }}{{ ransomware_path }}{{ \
     ransomware_name }}"
7          dest: "{{ remote_windows_user_directory }}{{ \
     ransomware_name }}"
8
9      - name: Trasferisci playbook interno
10       ansible.windows.win_copy:
11         src: "{{ rf_fulldisk_dir }}{{ \
     local_working_directory }}{{ internal_playbook_name }}"
12         dest: "{{ remote_windows_user_directory }}{{ \
     internal_playbook_name }}"
13
14     - name: Trasferisci inventario Ansible
15       ansible.windows.win_copy:
16         src: "{{ rf_fulldisk_dir }}{{ \
     local_working_directory }}{{ internal_inventory_name }}"
```

```
17          dest: "{{ remote_windows_user_directory }}{{ \
    internal_inventory_name }}"
18
19      - name: Trasferisci file variabili ansible
20        ansible.windows.win_copy:
21          src: "{{ rf_fulldisk_dir }}{{ \
    local_working_directory }}{{ ansible_variables_name }}"
22          dest: "{{ remote_windows_user_directory }}{{ \
    ansible_variables_name }}"
23
24      - name: Avvia playbook interno
25        ansible.windows.win_command: wsl ansible-playbook -i \
    {{ remote_linux_user_directory }}{{ \
    internal_inventory_name }} {{ \
    remote_linux_user_directory }}{{ internal_playbook_name \
    }} --extra-vars "@{{ remote_linux_user_directory }}{{ \
    ansible_variables_name }}"
26        async: 3600
27        poll: 0
28
29  - hosts: localhost
30    tasks:
31
32      - name: Attendi
33        ansible.builtin.pause:
34          seconds: "{{ attesa_spegnimento }}"
35
36      - name: Spegni VM
37        ansible.builtin.command: 'curl -k -X POST "\
    https://indirizzo_proxmox:8006/api2/json/nodes/{{ pve \
    }}/qemu/{{ vm_id }}/status/stop" -H "Authorization: \
    PVEAPIToken=utente@pam!nome_token=token"'
```

- **ext_inventory**
  L'inventario serve per indicare gli indirizzi degli host e altre informazioni che li riguardano, come le credenziali o il tipo di connessione da usare.

```
1  [windows]
2  host ansible_host=indirizzo.ip.della.vm
3
4  [windows:vars]
5  ansible_user=username
6  ansible_password=password
7  ansible_port=5985
8  ansible_connection=winrm
9  ansible_winrm_transport=basic
10 ansible_winrm_server_cert_validation=ignore
```

```
ansible_winrm_operation_timeout_sec= 800
ansible_winrm_read_timeout_sec=900
```

- **internal_playbook.yml**
  Una volta trasferiti tutti i file viene eseguito il playbook interno, che disabilita la scheda di rete (per ottenere l'air-gap), decomprime il ransomware, lancia il file batch per riempire il disco e come ultimo passo esegue il ransomware.

```
- hosts: windows
  tasks:

  - name: Stacca scheda di rete
    community.windows.win_net_adapter_feature:
      interface: '*'
      state: disabled
      component_id:
        - ms_tcpip
        - ms_tcpip6
        - ms_implat
        - ms_lltdio
        - ms_rspndr
        - ms_lldp
        - ms_msclient
        - ms_pacer

  - name: Decomprimi ransomware
    community.windows.win_unzip:
      src: "{{ remote_windows_user_directory }}{{ \
     ransomware_name }}"
      dest: "{{ ransomware_dest_path }}"
      delete_archive: false

  - name: Riempi disco con ranflood
    ansible.windows.win_command:
      argv:
        - "{{ ranflood_batch }}"

  - name: Avvia ransomware
    ansible.windows.win_command:
      argv:
        - "{{ ransomware_exec_path }}"
    async: 1800
    poll: 0
```

- **internal_inventory**
  L'inventario interno ha le stesse funzionalità e struttura di quello esterno, con la differenza che l'indirizzo è impostato a 127.0.0.1, l'interfaccia di loopback, per far collegare Ansible alla macchina stessa su cui è in esecuzione il playbook.
- **variables.yml**
  Con Ansible, si possono definire delle variabili in un file aggiuntivo per avere un'organizzazione migliore.
  In *variables.yml* troviamo tutte le variabili a cui si fa riferimento nei playbook, ad esempio *rf_fulldisk_dir*, *ransomware_name* o *internal_inventory_name*.

5.2 Memoria disponibile e file batch

Come spiegato in precedenza, le macchine virtuali sono state predisposte per i test avendo già memoria libera limitata.
Tuttavia, non è possibile occupare completamente lo spazio perché, ai fini delle prove, si devono svolgere operazioni come la copia dei file per Ansible e dell'archivio contenente il ransomware (che non sempre è lo stesso); inoltre anche la connessione e il funzionamento stessi di Ansible richiedono la creazione di alcuni file temporanei sulla macchina host, seppur di dimensioni ridotte.

Per risolvere questo problema entra in gioco lo script batch, che si avvale dell'uso di Ranflood per occupare la memoria disponibile rimanente in base a questi passaggi:

1. Avvia demone di Ranflood
2. Avvia il flooding con il relativo comando del client
3. Controlla ciclicamente che lo spazio disponibile sia inferiore ad 1MB
4. Quando la condizione è vera, ferma il flooding col comando del client
5. Ferma il demone di Ranflood

Entrando più nel dettaglio, la specifica implementazione dello script è la seguente:

```
1 @echo off
2 setlocal enabledelayedexpansion
3
4 :: Passo 1a: Avvia il demone di Ranflood
5 start "" C:\path\to\ranfloodd C:\path\to\settings.ini
6
7 :: Passo 1b: Aspetta qualche secondo in modo che il demone sia \
     pronto
8 timeout /t 3 /nobreak
9
```

```
:: Passo 2: Avvia il task di flooding
C:\path\to\ranflood flood start random C:\path\to\target-folder

:: Passo 3: Usa il comando list del client per ottenere l'ID del \
   task
C:\path\to\ranflood flood list > C:\path\to\task_list.log

:: Passo 4: Estrai l'ID del task dall'output del comando list
:: Estrai l'ultima riga
set "taskID="
for /f "tokens=* delims=" %%A in (C:\path\to\task_list.log) do (
    set "lastLine=%%A"
)

:: Estrai l'ultimo toke, ovvero l'ID del task
for /f "tokens=3 delims=| " %%A in ("!lastLine!") do (
    set "taskID=%%A"
)

:: Passo 5: Controlla lo spazio, se inferiore a 1MB ferma il \
   flooding usando l'ID
:check_space
for /f "tokens=*" %%A in ('powershell -command "$drive = \
   Get-PSDrive -Name C; $drive.Free"') do (
    set "freeBytes=%%A"
)
set /a freeKB=!freeBytes!/1024

if !freeKB! geq 1024 (
    timeout /t 2 /nobreak
    goto check_space
)

C:\path\to\ranflood flood stop random %taskID%

:: Passo 7: Ferma il demone
taskkill /f /im ranfloodd.exe

endlocal
```

5.3 Analisi del disco

Prima di poter controllare l'esito delle prove, è fondamentale salvare i checksum dei file presenti nella macchina senza che vi sia stato eseguito il ransomware.
Essendo le VM uguali, poiché clonate dallo stesso template, è sufficiente svolgere

questa operazione una volta, usando il comando apposito di Filechecker, in cui vanno indicati il percorso dove salvare i checksum e quello della cartella di cui controllare i file:

```
java -jar filechecker.jar save /path/to/checksum /path/to/save
```

Fatto questo, si può procedere con l'analisi sulle VM "infette".
Dopo l'esecuzione del playbook, il disco viene trasferito su un'altra macchina virtuale.
Questa ha come sistema operativo Ubuntu e, dopo aver montato il disco, tramite il comando check di Filechecker si controlla il numero di file ancora integri e il numero di quelli criptati dal virus.

Il comando può essere eseguito in questo modo:

```
java -jar filechecker.jar check /path/to/checksum /path/to/report /path/to/check
```

Il primo argomento è il file dei checksum precedentemente generato col comando save, poi abbiamo il percorso in cui salvare il report e infine la cartella su cui effettuare il controllo.

Capitolo 6

Analisi dei risultati

Il report generato da Filechecker è un file JSON [Cro] contenente i percorsi di tutti i file non criptati (o recuperati grazie a una copia), e relativi checksum, tra quelli specificati nel file prodotto dal comando save.
Si può intuire che analizzare direttamente questi report non è un metodo praticabile per comprendere e intepretare i dati.

Per ottenere informazioni più facilmente interpretabili, impieghiamo il *report analyser*, uno strumento sviluppato con Ruby [Tea] che, dati dei report in input, genera dei grafici con le informazioni rilevanti estrapolate dai file JSON.
I dati riportati sono degli aggregati dei vari test, in quanto, come spiegato precedentemente, per ogni campione sono state condotte quattro prove.

6.1 Risultati generali

Tutti i ransomware usati nei test hanno criptato dei file, anche se in misura irrisoria in confronto al totale, infatti si va dallo 0,2% di file criptati di Winlocker e Birele ad un massimo pari a 0,7% con Annabelle.
La scarsa attività osservata potrebbe essere dovuta alla mancanza di disponibilità di memoria, anche se, considerate le percentuali di file criptati ($\geq$ 0,2%) e la dimensione della cartella analizzata (23GB), ovvero quella dell'utente, ci si può accorgere che anche i campioni meno attivi hanno cifrato file per uno spazio totale di almeno 45MB, molto di più di quello che era rimasto disponibile. Questo dato sembrerebbe indicare che tutti gli esemplari utilizzino la sovrascrittura durante le loro operazioni, senza necessitare di ulteriore spazio su disco.

I file rimasti integri dopo l'azione dei ransomware sono spesso molto simili, quasi identici, ma non è necessariamente un'informazione significativa poiché, considerato

l'impatto minimale esercitato dai virus, è naturale che sui file "sani", essendo in quantità molto più elevata, non ci siano grandi differenze.

Le estensioni dei file persi sono più interessanti e vi si nota maggiore eterogeneità: nonostante le tipologie attaccate dai virus siano bene o male le stesse, le percentuali in cui vengono colpite sono diverse.
In particolare risaltano WannaCry e Annabelle, che differenziandosi dagli altri esemplari, vanno a criptare maggiormente i .jpg, rispettivamente col 39,3% e 55,7%.

La cartella con il numero maggiore di file lasciati inalterati è AppData in tutti i casi, con una percentuale sempre superiore al 96%, anche se, come spiegato sopra, non è un dato significativo a causa della piccola quantità di file cifrati, e questo dipende più dalla distribuzione originale dei file.
È importante notare che AppData non è il percorso che occupa più memoria; al contrario non rappresenta neanche il 15% dello spazio (3,3GB), ma contiene un elevato numero di file di piccole dimensioni, il che lo porta al primo posto in questo ambito.

Osservando le cartelle più interessate dall'azione dei virus, notiamo come nuovamente WannaCry e Annabelle si distinguono, concentrandosi più degli altri ransomware sulla cartella Documents, con il 39,2% e 55,7%.
I restanti campioni portano avanti quasi tutte le loro operazioni in AppData, confermando quanto esposto riguardo i percorsi con più file intatti.

6.2 Pattern comuni

Oltre alle considerazioni generali, analizzando i grafici, è possibile individuare dei comportamenti comuni ad alcuni ransomware.

6.2.1 Annabelle e WannaCry

Uno dei pattern più evidenti è tra Annabelle e WannaCry, che risalta data la netta differenza che presentano rispetto gli altri quattro campioni.

Sono i primi nella classifica di file cripati, anche se WannaCry è più in linea con i restanti ransomware sotto questo punto di vista.
Entrambi colpiscono in maniera similare i file, in particolare interessandosi alle immagini (.jpg), col 55,7% di Annabelle e 39,3% di WannaCry, cosa che non si è riscontrata in nessun altro esemplare. Inoltre si concentrano molto di più sulla cartella Documents rispetto agli ulteriori campioni, operando in questo percorso rispettivamente per il 55,7% e 39,2%.

6.2.2 Birele e Winlocker

Un secondo pattern notabile è tra Birele e Winlocker, che hanno criptato lo 0,2% dei file.
In questo caso è meno evidente del precedente, ma controllando i dati, si nota la correlazione.

Le estensioni dei file colpiti e le loro percentuali sono estremamente simili, con differenze irrisorie.
La cartella in cui hanno agito principalmente è AppData, con 99,1% dei file criptati per ognuno dei due.

6.2.3 7ev3n e Fantom

L'ultima coppia individuata in base alla metodologia di azione è composta da 7ev3n e Fantom.

Entrambi questi ransomware hanno agito sullo 0,3% dei file totali e hanno un pattern simile per quanto riguarda i file colpiti.
Al primo posto, con buon margine, troviamo per entrambi i file senza alcuna estensione, seguiti dai file di testo.

In merito alle cartelle che hanno subito l'attacco, troviamo ancora AppData al 99,5%, analogamente alla coppia precedente, ma con un piccolo incremento dello 0,4%.

Riassumendo, i due esemplari a differenziarsi particolarmente restano Annabelle e WannaCry, che mostrano un'attività molto differente e sembrano anche andare a ricercare con più tenacia certi tipi di file e percorsi sul disco. Un'importante precisazione da fare in relazione a quanto appena detto è che la concentrazione dell'azione di questi due ransomware su alcuni tipi di file potrebbe dipendere dalla propensione ad agire nella cartella Documents, in cui sono presenti anche delle immagini, e meno in AppData, dove si sono svolte invece la maggior parte delle operazioni di 7ev3n, Birele, Fantom e Winlocker, che non vi hanno potuto trovare file .jpg.

	Estensioni		Cartelle	
Totale	File intatti	File criptati	File intatti	File criptati

Tabella 6.1: Grafici dei risultati

File intatti
File criptati

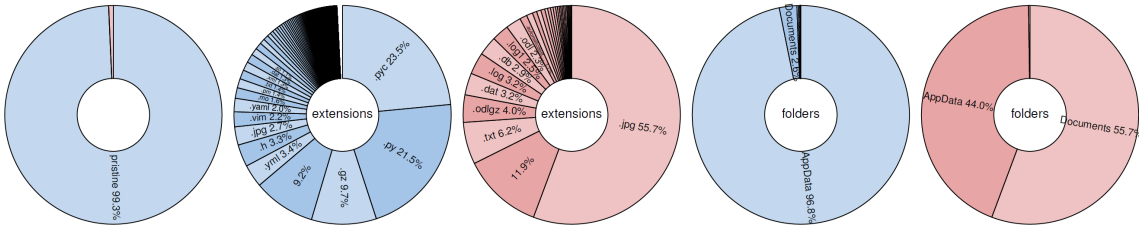


Figura 6.1: Risultati con Annabelle

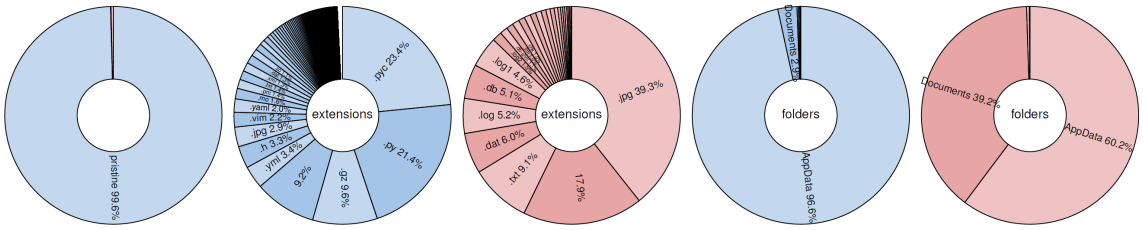


Figura 6.2: Risultati con WannaCry

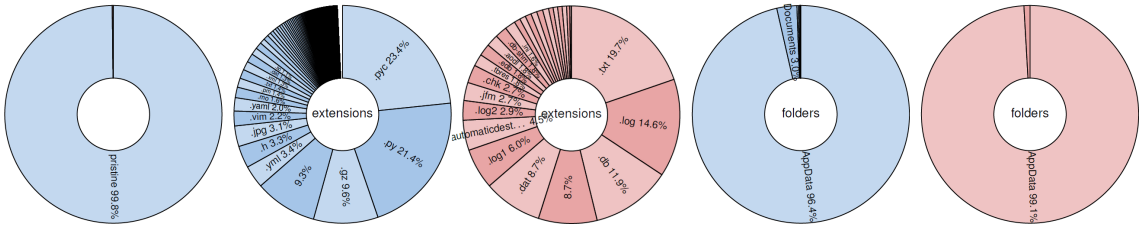


Figura 6.3: Risultati con Birele

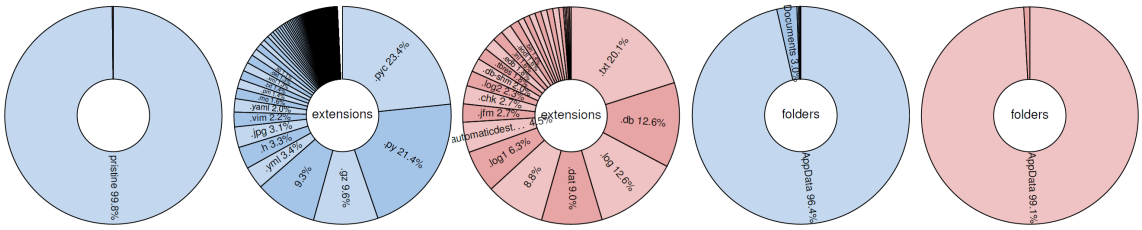


Figura 6.4: Risultati con Winlocker

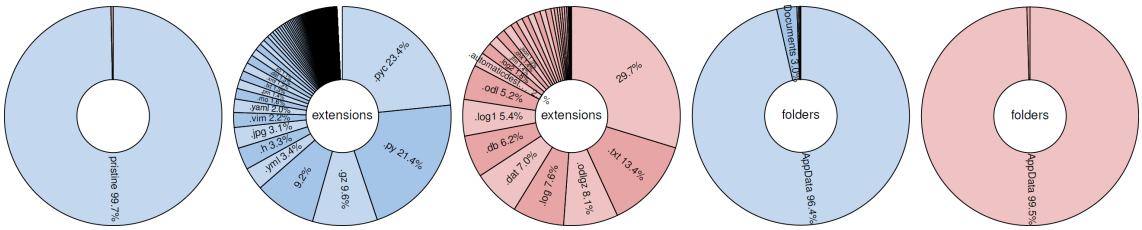


Figura 6.5: Risultati con 7ev3n

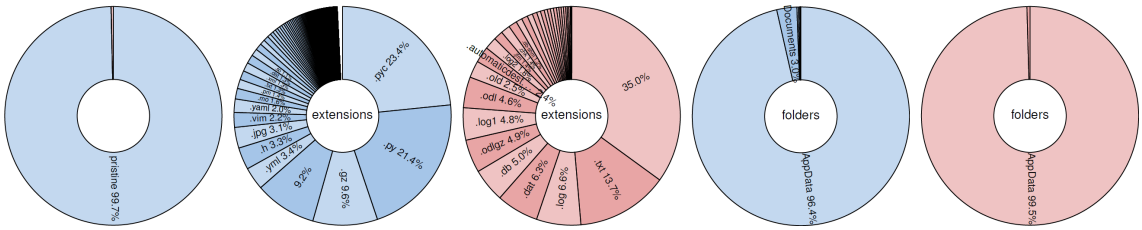


Figura 6.6: Risultati con Fantom

Capitolo 7

Conclusioni

L'analisi condotta sui comportamenti dei ransomware in condizioni di spazio su disco limitato ha fornito risultati interessanti e, in alcuni casi, inaspettati.
Nonostante la ridotta disponibilità di memoria, tutti i ransomware testati sono stati in grado di criptare una piccola percentuale di file, confermando la capacità di continuare le loro operazioni senza necessitare di spazio aggiuntivo sul disco, presumibilmente attraverso la sovrascrittura dei file esistenti. Questo aspetto dimostra che i dati in memoria possono essere corrotti in mancanza di spazio libero, quindi sono necessari dei meccanismi di difesa anche in questa casistica.

Uno dei principali risultati emersi riguarda la distribuzione delle operazioni di criptazione tra le varie tipologie di file e cartelle.
In particolare, Annabelle e WannaCry hanno mostrato un comportamento distintivo rispetto agli altri campioni. Questi ransomware si sono concentrati maggiormente su immagini, nello specifico i .jpg, e sulla cartella Documents, suggerendo una preferenza per file personali e sensibili per l'utente, che potrebbe indicare che questi ransomware siano stati progettati con l'intento di massimizzare il danno inflitto all'utente colpendo file che contengono informazioni di maggiore valore emotivo o pratico.
Al contrario, ransomware come Birele, Winlocker, 7ev3n, e Fantom hanno mostrato una preferenza per la cartella AppData, colpendo principalmente file tecnici o di sistema, come .log, .dat e altri file senza estensione. In questo modo tendono a criptare file meno critici per l'utente ma comunque rilevanti per il sistema.

L'identificazione di pattern comuni tra i diversi ransomware ha rivelato la presenza di "famiglie" di comportamento. Annabelle e WannaCry si sono distinti come ransomware più aggressivi e orientati al danneggiamento di file personali, mentre Birele, Winlocker, 7ev3n e Fantom hanno mostrato un approccio diverso, concentrandosi

su file di sistema, seppur con delle differenze sui tipi di file colpiti tra la prima e la seconda coppia.

Nonostante la quantità di dati raccolti, la piccola percentuale di file criptati in questo studio potrebbe limitare la possibilità di trarre conclusioni definitive sul comportamento dei ransomware in contesti reali.
Tuttavia, i risultati ottenuti mostrano chiaramente che, anche in presenza di un disco pieno, i ransomware possono continuare a operare e criptare file, spesso in modo mirato. Le differenze nei tipi di file e cartelle colpiti evidenziano come i ransomware non siano tutti uguali, ma anzi possano adottare strategie diverse a seconda delle loro caratteristiche.

7.1 Considerazioni finali

Questa ricerca ha dimostrato che lo spazio disponibile su disco non rappresenta una barriera per l'azione dei ransomware, i quali possono continuare a criptare file sovrascrivendo quelli esistenti. Inoltre, ha messo in evidenza come alcuni ransomware si concentrino su file personali e cartelle sensibili, mentre altri preferiscano colpire aree più tecniche del sistema.

Questi risultati ci permettono di capire che futuri sistemi di sicurezza dovranno tenere conto delle diverse strategie di attacco adottate dai ransomware, ad esempio implementando un approccio dinamico nella fase di difesa e mitigazione degli effetti del virus. Questo studio può quindi rappresentare un punto di partenza per ulteriori ricerche sul comportamento dei ransomware in condizioni diverse, e per lo sviluppo di contromisure più efficaci che siano in grado di contrastare le specifiche tecniche adottate da questi malware.

Bibliografia

[ArMS18] Bander Ali Saleh Al-rimy, Mohd Aizaini Maarof, and Syed Zainudeen Mohd Shaid. Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions. *Computers & Security*, 2018.

[BGM+] Davide Berardi, Saverio Giallorenzo, Andrea Melis, Simone Melloni, and Marco Prandini. Ranflood. GitHub repository.

[BGM+23] Davide Berardi, Saverio Giallorenzo, Andrea Melis, Simone Melloni, Loris Onori, and Marco Prandini. Data flooding against ransomware: Concepts and implementations. *Computers & Security*, 131:103295, 2023.

[BGM+24] Davide Berardi, Saverio Giallorenzo, Andrea Melis, Simone Melloni, and Marco Prandini. Ranflood: A mitigation tool based on the principles of data flooding against ransomware. *SoftwareX*, 25:101605, 2024.

[Com23] Tommaso Compagnucci. Design e sviluppo di un framework scalabile e isolato per l'indagine automatizzata sui ransomware. Bachelor's thesis, Alma Mater Studiorum - Università di Bologna, 2023.

[Cro] Douglas Crockford. The JSON Data Interchange Format. [Consultato il: 2024-10-18].

[GAF+15] André Ricardo Abed Grégio, Vitor Monte Afonso, Dario Simões Fernandes Filho, Paulo Lício de Geus, and Mario Jino. Toward a Taxonomy of Malware Behaviors. *The Computer Journal*, 2015.

[Gay] Ayush Gayakwad. Malware Database. GitHub repository.

[KAM+16] Amin Kharaz, Sajjad Arshad, Collin Mulliner, William Robertson, and Engin Kirda. Unveil: A large-scale, automated approach to detecting ransomware. In *25th USENIX Security Symposium (USENIX Security 16)*, 2016.

[Mic] Microsoft. What is the Windows Subsystem for Linux? [Consultato il: 2024-10-8].

[OALU22] Harun Oz, Ahmet Aris, Albert Levi, and A. Selcuk Uluagac. A Survey on Ransomware: Evolution, Taxonomy, and Defense Solutions. *ACM Computing Surveys*, 2022.

[Tea] Ruby Core Team. Il linguaggio di programmazione Ruby. [Consultato il: 2024-10-18].

[TML+20] Fei Tanga, Boyang Maa, Jinku Li, Fengwei Zhang, Jipeng Su, and Jianfeng Ma. RansomSpector: An introspection-based approach to detect crypto ransomware. *Computers & Security*, 2020.

[Wika] Wikipedia. Computer cluster. [Consultato il: 2024-10-12].

[Wikb] Wikipedia. Secure copy protocol. [Consultato il: 2024-10-12].

[WJC+19] Peiying Wang, Shijie Jia, Bo Chen, Luning Xia, and Peng Liu. MimosaFTL: Adding Secure and Practical Ransomware Defense Strategy to Flash Translation Layer. In *Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy*, 2019.

[yti] Thezoo. GitHub repository.