Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Informatica

Rilevazione di Ransomware attraverso Leaky Bucket

Relatore:
Chiar.mo Prof.
Saverio Giallorenzo
Correlatore:
Dott.
Simone Melloni

Presentata da:
Alessandro Maria
Benassi Trenta

Sessione III
A.A. 2023/2024

Indice

Capitolo 1

Introduzione

Il crescente impatto delle minacce informatiche, in particolare i ransomware, nelle infrastrutture IT moderne ha spinto alla ricerca e allo sviluppo di soluzioni di sicurezza sempre più avanzate e sofisticate. Queste minacce non solo causano gravi interruzioni operative, ma comportano anche significativi costi economici e danni alla reputazione per le entità colpite.

In questo contesto, il rilevamento tempestivo e affidabile del ransomware diventa una priorità critica per la sicurezza informatica.

Uno dei tipi di ransomware più diffusi, focus di questa tesi, cripta i file di un utente, rendendoli inaccessibili, e richiede un riscatto per la chiave di decrittazione. La perdita, ad esempio, di dati onerosi da ottenere può spingere una azienda a cedere a ricatti e pagare cyber criminali perché meno costoso rispetto a riacquisire i dati. Data la natura distruttiva dei ransomware e la loro rapida evoluzione, è fondamentale sviluppare metodi di rilevamento che siano sia efficaci che efficienti, capaci di adattarsi alle nuove strategie adottate dagli attaccanti.

Questo progetto di tesi propone lo sviluppo di un nuovo approccio per il rilevamento del ransomware basato sull'uso dell'algoritmo Leaky Bucket. Questo metodo è tradizionalmente impiegato per il controllo della congestione nella gestione del traffico di rete, ma la sua applicazio-

ne nel contesto della sicurezza informatica offre un potenziale inesplorato. L'obiettivo è di esaminare se l'algoritmo Leaky Bucket, adattato per il monitoraggio delle attività sospette nei sistemi, possa effettivamente identificare e limitare le operazioni di ransomware prima che causino danni irreparabili. La tesi analizza la teoria dietro l'algoritmo Leaky Bucket, adattandola per rilevare modelli di traffico anomali che potrebbero indicare la presenza di ransomware.

Questo algoritmo ha un funzionamento comparabile ad un secchio che, bucato, che perde acqua. Inoltre, viene proposta un'implementazione di questo metodo e test di rilevazione contro ransomware reali in un ambiente controllato, utilizzando macchine virtuali per studiare gli attacchi in scenari realistici. Attraverso questo approccio sperimentale, la ricerca mira a valutare l'efficacia del Leaky Bucket come strumento di rilevamento, con l'obiettivo di integrarlo in sistemi di sicurezza più ampi per migliorare la resilienza contro le minacce di ransomware.

Capitolo 2

Background Tecnico

Prima di entrare nel vivo dell'implementazione, è importante fissare alcune definizioni sulle tecnologie utilizzate.

2.1 Shannon Entropy

L'entropia di Shannon è un concetto fondamentale nella teoria dell'informazione, introdotto dal matematico e ingegnere Claude Shannon[1]. Essa misura la quantità di incertezza o informazione contenuta in un messaggio o in una fonte di dati. Essenzialmente, è la misura della predicibilità di un qualsiasi byte specifico in un messaggio, basandosi sui byte precedenti.

Un file con una struttura prevedibile o sequenze di bit che vengono ripetute frequentemente si dice abbia una entropia bassa, mentre file in cui il prossimo byte è relativamente indipendente dal valore dei byte precedenti si dice abbia una entropia alta.

Matematicamente, l'entropia di Shannon $H(X)$ (misurata in bit) per un messaggio è definita come:

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log_2(P(x_i)) \tag{2.1}$$

Dove n è il numero di byte del messaggio e $P(X_i)$ è la probabilità che il byte i appaia nella sequenza di byte. Il tutto ha il segno meno davanti in modo che il risultato sia positivo o zero.

2.2 Leaky Bucket

Con Leaky Bucket ("secchio che perde") ci si riferisce ad un modello per descrivere sistemi in cui ci sono risorse che vengono ridotte/smaltite ad intervalli regolari (solitamente di tempo), e che possono diventare troppe da contenere.

I componenti base di un leaky bucket sono:

- Capienza: quante risorse si possono contenere prima che il secchio strabordi ("overflow").
- Finestra di perdita: quanto è grande l'intervallo ogni quanto le risorse vengono smaltite.
- Quantità dello smaltimento: quante risorse vengono smaltite una volta arrivato il momento.

Ad esempio si può pensare ad un buco in una nave:

- se entra più di una certa quantità di acqua, la nave affonda.
- una persona può gettare fuori dalla nave dell'acqua con uno strumento (ad esempio una tazza o un secchio), ma per farlo impiega un certo tempo.
- ogni volta che questa persona completa l'azione di gettare fuori dell'acqua, la quantità è discreta (e virtualmente approssimabile alla capienza dello strumento utilizzato).

2.3 Classi astratte e interfacce in Java

Poiché il linguaggio utilizzato nel progetto è Java, in questa sezione sarà usata la terminologia specifica di Java per introdurre le componenti del linguaggio, che saranno poi utilizzate per descrivere l'implementa-

zione. I concetti sono applicabili agli altri linguaggi di programmazione ad oggetti che supportano questo tipo di strutture dati, anche se la nomenclatura potrebbe essere leggermente diversa.

Una classe astratta è una classe che può includere o meno metodi astratti, non può essere istanziata, ma possono essere reificate da sottoclassi. Un metodo astratto è un metodo dichiarato senza implementazione, come questo:

```
abstract void moveTo(double deltaX, double deltaY);
```

Se una classe include metodi astratti, allora la classe stessa deve essere dichiarata come astratta, come in:

```
public abstract class GraphicObject {
   // dichiara i campi
   // dichiara i metodi non astratti
   // dichiara i metodi astratti come:
   abstract void draw();
}
```

Quando una classe astratta viene reificata da una sottoclasse, la sottoclasse di solito fornisce implementazioni per tutti i metodi astratti nella sua classe madre e, se non lo fa, anche la sottoclasse deve essere dichiarata come astratta. Con le classi astratte, è possibile dichiarare campi che non sono statici e finali e definire metodi concreti pubblici, protetti e privati. Con le interfacce, tutti i campi sono automaticamente public, static e final e tutti i metodi che si dichiarano o definiscono (come metodi predefiniti) sono pubblici. Inoltre, è possibile estendere una sola classe, sia essa astratta o meno, mentre è possibile implementare un numero qualsiasi di interfacce.

Quale scegliere, classi astratte o interfacce? Si può considerare l'utilizzo di classi astratte in uno di questi casi:

- Si desidera condividere il codice tra diverse classi strettamente cor-

relate.

- Ci si aspetta che le classi che estendono la classe astratta abbiano molti metodi o campi comuni, o richiedano modificatori di accesso diversi da public (come protected e private).
- Si desidera dichiarare campi non statici o non finali. Ciò consente di definire metodi che possono accedere e modificare lo stato dell'oggetto a cui appartengono

Si può considerare l'utilizzo di interfacce in uno di questi casi:

- Ci si aspetta che classi non correlate implementino l'interfaccia. Ad esempio, le interfacce "Comparable" e "Cloneable" sono implementate da molte classi non correlate.
- Si desidera specificare il comportamento di un particolare tipo di dati, ma non ci si preoccupa di chi implementa il suo comportamento.
- Si vuole sfruttare l'ereditarietà multipla del tipo

Un esempio di classe astratta nel JDK è AbstractMap, che fa parte del framework Collections. Le sue sottoclassi (che includono HashMap, TreeMape ConcurrentHashMap) condividono molti metodi (tra cui get, put, isEmpty, containsKeye, containsValue) definiti da AbstractMap.

Capitolo 3

Implementazione

Il progetto si divide in più componenti: AbstractBucket e sottoclassi, Watcher e PathHandler

3.1 AbstractBucket e sottoclassi

3.1.1 AbstractBucket

L'AbstractBucket è una classe astratta progettata per garantire l'integrità del valore del bucket, un intero che rappresenta la quantità di acqua presente in un dato momento, in ambienti multithreading.

Questo è particolarmente rilevante in scenari in cui si decide di introdurre un controllore di bucket globale, un componente non attualmente implementato, che potrebbe ottenere informazioni e gestire il singolo bucket. In tali situazioni, la classe è progettata per evitare la lettura di valori non aggiornati, un aspetto cruciale per la correttezza dell'applicazione.

La classe AbstractBucket contiene i campi visti in precedenza:

- capacity (Capacità): Rappresenta il limite massimo di "acqua" che il bucket può contenere. Questa è una soglia critica, poiché superarla indica una potenziale condizione di trabocco o overflow, indice di

un'anomalia o di un attacco imminente nel contesto di rilevamento del malware.

- leakWindow (Finestra di Perdita): Definisce l'intervallo, o unità astratta, dopo il quale avviene il leaking. Sebbene tradizionalmente si possa pensare a questa unità in termini di tempo (ad esempio, secondi), nel contesto di questa implementazione, essa rappresenta una unità astratta che verrà stabilita dalle sottoclassi; attualmente sono implementate due tipologie di perdita: sia a tempo sia a file esaminati. Questo intervallo è cruciale per determinare la frequenza con cui il bucket deve "perdere acqua" e regolare la sua capacità.
- leakAmount (Quantità di Perdita): Indica la quantità di "acqua" che il bucket perde in ogni evento di leaking. La dimensione di questa perdita è vitale per gestire l'overflow del bucket, assicurando che la capacità non venga superata in condizioni di normale esecuzione del sistema.

Il costruttore della classe AbstractBucket e i suoi metodi chiave hanno un ruolo fondamentale nella definizione del comportamento del bucket e nella sua capacità individuare minacce in tempo reale. Il costruttore della classe AbstractBucket ha il compito principale di inizializzare le variabili dell'istanza. Queste variabili includono la capacità del bucket, il valore corrente dell'acqua nel bucket, la dimensione della leakWindow e il leakAmount. Il costruttore accetta parametri specifici per ciascuna di queste variabili e li assegna alle rispettive proprietà dell'oggetto. Questa operazione garantisce che ogni istanza del bucket sia configurata con i parametri appropriati al momento della creazione.

Il metodo GetWater è una funzione accessorio che permette di consultare il volume corrente di "acqua" nel bucket. Questa funzione è essenziale per monitorare lo stato del bucket e per valutare se sono necessarie azioni correttive o preventive in base al volume di acqua, particolarmente utile in caso di gestori globali di bucket.

Il metodo AddWater accetta due parametri: il volume di acqua da

aggiungere e un insieme di metriche relative alle operazioni sui file, specificamente il numero di file eliminati, creati e modificati. Questo metodo inizia invocando BeforeAddWater, un metodo che può essere sovrascritto dalle classi derivate per eseguire operazioni preliminari prima dell'aggiunta dell'acqua. Dopo l'esecuzione di BeforeAddWater, AddWater procede ad aggiungere il volume specificato di acqua al bucket. Durante questo processo, il metodo effettua dei controlli per garantire che il bucket non superi la sua capacità massima. Se il volume di acqua dovesse causare un trabocco, viene invocato il metodo overflow.

Questo metodo è anch'esso progettato come un metodo che può essere sovrascritto dalle classi derivate per definire le azioni specifiche da intraprendere, come la generazione di allarmi o altre misure di sicurezza.

Le classi derivanti da Abstract Bucket sono attualmente organizzate in due categorie: una basata sul tempo e l'altra sul numero di file elaborati. Entrambe queste categorie sono classi astratte che, a loro volta, hanno classi derivate ulteriormente specializzate che rappresentano le implementazioni concrete e istanziabili del bucket. Queste "classi nipote" di Abstract Bucket sono personalizzate per gestire specifici scenari di uso e configurazioni operative.

Il metodo BeforeAddWater come detto prima è progettato come metodo predefinito vuoto all'interno della classe Abstract Bucket. Questo metodo offre un punto di intervento per le classi derivate, consentendo loro di eseguire operazioni preliminari prima dell'aggiunta effettiva dell'acqua al bucket. L'implementazione di default di questo metodo è intenzionalmente vuota, per fornire la massima flessibilità alle sottoclassi che possono avere esigenze specifiche non anticipate nella classe base, eventualmente lasciandolo vuoto.

Il metodo overflow è astratto nella classe Abstract Bucket e deve essere necessariamente implementato nelle classi derivate. Questo metodo definisce le azioni da intraprendere quando il bucket raggiunge o supera la sua capacità, ovvero quando si verifica un trabocco. L'implementazione

di questo metodo nelle classi figlie o nipoti può variare significativamente a seconda delle politiche di sicurezza specifiche, come l'attivazione di allarmi, l'avvio di procedure di mitigazione o altre risposte automatiche a situazioni critiche, o anche solo logging, come nel caso specifico implementato.

Il metodo leak è anch'esso astratto e rappresenta un elemento fondamentale nella gestione del bucket. Questo metodo deve essere implementato nelle classi derivate per definire la strategia di leaking specifica del bucket, che può variare a seconda della metrica di controllo utilizzata (tempo o numero di file). L'implementazione di leak può comprendere, ad esempio, la diminuzione periodica del volume di acqua nel bucket in funzione del tempo o del numero di file processati, in linea con la politica di gestione stabilita.

Le sottoclassi dirette di AbstractBucket sono specificate per operare secondo due principali dimensioni di controllo: il tempo, TimeBucket, e il numero di file modificati (FileBucket). Queste classi astratte verranno ulteriormente specializzate attraverso classi figlie concrete che definiscono in modo preciso come i metodi leak e overflow devono comportarsi in scenari reali di utilizzo.

3.1.2 TimeBucket

Nella classe TimeBucket, sottoclasse diretta di AbstractBucket, il processo di gestione dell'acqua è guidato principalmente dalla misurazione del tempo. Questa classe utilizza il tempo come criterio fondamentale per determinare quando e quanto perdere acqua tramite leak prima di aggiungere nuovo volume. Questa classe contiene una variabile chiamata lastLeakTimestamp che contiene il timestamp dell'ultima perdita.

Il processo può essere suddiviso nei seguenti passaggi chiave che definiscono l'implementazione del metodo leak:

1. Calcolo del Tempo Trascorso: Prima di procedere all'aggiunta di nuova acqua, il bucket calcola il tempo trascorso dall'ultimo evento

di leaking. Questa misurazione è facilitata da una variabile che tiene traccia dell'ultimo timestamp di leaking.

2. Determinazione delle Perdite: Utilizzando il tempo calcolato, il bucket determina quante perdite avrebbero dovuto verificarsi dall'ultimo evento di leaking. Questo calcolo si basa su una formula predefinita che considera la frequenza di leaking in funzione del tempo.
3. Applicazione delle Perdite: Una volta determinato il volume di acqua da perdere, il bucket esegue l'azione di leaking, riducendo il suo contenuto di acqua di conseguenza. Poiché il bucket non può avere acqua negativa, se il risultato diventa minore di zero viene reimpostato a zero.
4. Aggiornamento del Timestamp: Dopo l'evento di leaking, il bucket aggiorna il timestamp dell'ultimo leaking. Questo aggiornamento è essenziale per garantire che il calcolo del tempo per il prossimo evento di leaking sia accurato.
5. Aggiunta di Acqua: Successivamente, il bucket procede all'aggiunta dell'acqua. Questo passaggio segue il leaking per assicurare che qualsiasi nuovo volume aggiunto non superi la capacità del bucket immediatamente dopo un evento di perdita, causanda falsi positivi.

Il metodo BeforeAddWater chiama semplicemente il metodo leak con il timestamp corrente.

3.1.3 FileBucket

Nella classe FileBucket, che è anch'essa una sottoclasse diretta di AbstractBucket, il processo di gestione dell'acqua è guidato principalmente dalla misurazione dei file modificati, divisi in tre tipologie: rimozione, creazione e modifica del contenuto.

Questa classe utilizza la somma totale del numero di queste tre liste come criterio fondamentale per determinare quando e quanto perdere prima di aggiungere nuovo volume; inoltre mantiene un contatore interno

chiamato fileCount, che rappresenta il numero di files processati, questo contatore viene resettato ogni volta che avviene una perdita, e si comporta quindi in modo simile al timestamp dell'ultima perdita per il TimeBucket.

Il processo può essere suddiviso nei seguenti passaggi chiave che definiscono l'implementazione del metodo leak:

1. Calcolo del numero di files "passati": Prima di procedere all'aggiunta di nuova acqua, il bucket calcola il numero di file modificati e lo aggiunge al contatore interno.
2. Determinazione delle Perdite: Utilizzando il counter interno, il bucket determina quante perdite avrebbero dovuto verificarsi dall'ultimo evento di leaking. Questo calcolo si basa su una formula predefinita che considera la frequenza di leaking in funzione del numero di file modificati.
3. Aggiornamento del counter: si aggiorna il contatore interno in base a quanti file sono stati modificati dopo la perdita corrente, considerando che questo valore non può superare la leakWindow (il nuovo valore sarà il resto della divisione intera tra il totale di file passati calcolato prima e il valore di leakWindow).
4. Applicazione delle Perdite: Una volta determinato il volume di acqua da perdere, il bucket esegue l'azione di leaking, riducendo il suo contenuto di acqua di conseguenza. Poiché il bucket non può avere acqua negativa, se il risultato diventa minore di zero viene reimpostato a zero.
5. Aggiunta di Acqua: Successivamente, il bucket procede all'aggiunta dell'acqua.

Il metodo BeforeAddWater calcola il numero di file totali che sono stati modificati in qualche modo, che sono divisi per tipo di modifica (rimozione, creazione, modifica di contenuto), e chiama il metodo leak con questo numero.

3.1.4 Classi Nipoti

Sia TimeBucket che FileBucket sono classi astratte che non implementano direttamente il metodo overflow. Questa mancanza di implementazione è intenzionale, poiché permette la creazione di sottoclassi più specializzate che definiscono specifiche azioni di trabocco in risposta a condizioni anormali rilevate. Esempi di tali sottoclassi includono SimpleTimeBucket e SimpleFileBucket, che implementano logiche specifiche di trabocco per gestire situazioni di errore in modo personalizzato.

Per questo progetto, in caso di rilevazione di anomalie o errori, SimpleTimeBucket e SimpleFileBucket sono configurati per registrare questi eventi in file di log. Questo metodo di segnalazione è essenziale per documentare e analizzare le attività sospette, permettendo agli operatori di sicurezza di intervenire prontamente e di adottare misure correttive o preventive basate sulle informazioni raccolte.

Le classi SimpleFileBucket e SimpleTimeBucket rappresentano le implementazioni concrete e istanziabili delle classi astratte FileBucket e TimeBucket rispettivamente. Queste classi finale sono progettate per operare nel sistema di rilevamento malware, con la specifica funzionalità di gestire le condizioni di trabocco in maniera personalizzata.

Una delle caratteristiche distintive di SimpleFileBucket e SimpleTimeBucket è l'implementazione personalizzata del metodo overflow. In queste classi, la sostituzione di overflow è progettato per registrare gli eventi di trabocco in modo dettagliato, scrivendo su file di log specifici. Questa registrazione include informazioni critiche come il nome del bucket, il momento preciso del trabocco, e la quantità di "acqua" nel bucket al momento dell'evento. Queste informazioni sono vitali per il monitoraggio e l'analisi post-evento, facilitando la comprensione delle dinamiche che hanno portato al trabocco.

Un altro aspetto importante di queste classi è la definizione dei file di log. Gli eventi di trabocco sono sono scritti in maniera uguale in numerose locazioni del sistema per evitare che, in caso di attacco ransomware,

tutte le informazioni siano cifrate e per evitare di perdere le informazioni sul malware individuato.

3.2 Watcher

La classe Watcher è concepita per monitorare i cambiamenti nei file system utilizzando la libreria Java.NIO2, agendo come contesto di esecuzione intorno alla funzionalità base di FileWatcher fornita dalla libreria. Questa classe è strutturata per essere eseguita come un thread, implementando l'interfaccia Runnable, e possiede un meccanismo di terminazione controllato tramite una variabile che segnala la necessità di concludere l'esecuzione.

Il metodo principale, run, contiene un ciclo `while(appIsRunning)`, ovvero continua finché la variabile di terminazione appIsRunning non viene impostata true esternamente. Durante ogni iterazione del ciclo, la classe Watcher esegue le seguenti operazioni principali:

1. Registrazione degli Eventi: Gli eventi di file come creazioni, modifiche e cancellazioni vengono ricevuti con un breve ritardo (tipicamente 500 millisecondi) per minimizzare l'overhead del monitoraggio in tempo reale. La classe Watcher unifica gli eventi per ridurre la duplicazione, ad esempio, trattando una serie di eventi di creazione seguiti da una modifica come un singolo evento di modifica.
2. Elaborazione degli Eventi: Quando gli eventi vengono rilevati, Watcher utilizza una funzione specificata nella FunctionalInterface Listener. Questa interfaccia è responsabile di definire come gli eventi sui file (creazione, modifica, cancellazione) debbano essere gestiti. La funzione associata al Listener viene invocata con gli insiemi di nomi dei file modificati, permettendo di eseguire azioni personalizzate basate su questi cambiamenti.

Questa configurazione rende Watcher versatile, capace di adattarsi a vari scenari d'uso, dove il monitoraggio in tempo reale dei file è cruciale.

La possibilità di specificare azioni personalizzate tramite la FunctionalInterface Listener permette agli sviluppatori di integrare il Watcher in sistemi più grandi, dove azioni automatizzate basate sui cambiamenti dei file possono innescare catene di eventi complesse all'interno delle applicazioni.

La classe Watcher contiene un metodo pubblico che le consente di registrare le cartelle da osservare, queste cartelle sono monitorate ricorsivamente.

Inoltre il campo handler contiene una referenza al PathHandler che usa questo Watcher, in modo che eventuali modifiche ad i suoi campi siano sempre rispettate senza bisogno di chiamate aggiuntive.

3.3 PathHandler

La classe PathHandler funge da coordinatore tra un Watcher e un Bucket, facendo da ponte tra il monitoraggio dei file e la gestione della capacità di risposta del sistema. Il ruolo di PathHandler è di orchestrare l'interazione tra questi due componenti chiave, assicurando che gli eventi rilevati dal Watcher siano adeguatamente gestiti attraverso aggiornamenti pertinenti al Bucket. PathHandler è progettato per essere eseguito su un thread dedicato, permettendo così un'elaborazione asincrona e isolata che migliora le prestazioni e la gestione delle risorse.

Le principali funzionalità e operazioni di questa classe sono:

- Integrazione Watcher-Bucket: All'interno del suo flusso di esecuzione, PathHandler inizializza e gestisce un'istanza di Watcher, che è configurato per monitorare specifiche cartelle. I cambiamenti rilevati sono comunicati al Bucket associato tramite una funzione listener personalizzata, che aggiorna il bucket in base alla natura degli eventi osservati (es. aggiungendo "acqua" in risposta a modifiche di file).

- Gestione degli Eventi: La funzione listener è responsabile per determinare come i segnali dal Watcher influenzino lo stato del Bucket. Questo meccanismo di feedback è vitale per mantenere il sistema aggiornato rispetto alle attività nel filesystem monitorato. Questa è la funzione nominata prima nel Watcher, e viene chiamata durante da gestione dei file.
- Controllo del Flusso di Esecuzione: PathHandler include metodi come stop o shutdown, che permettono di interrompere il monitoraggio e le operazioni del bucket. Questi metodi facilitano una gestione controllata del ciclo di vita del PathHandler, assicurando che le risorse possano essere liberate in modo appropriato quando non sono più necessarie. Questo offre un'interfaccia unificata al controllo di bucket e Watcher, senza dover interagire con ognuno separatamente.
- Configurabilità del Listener: Il Listener può essere definito staticamente in una sottoclasse di PathHandler o configurato dinamicamente tramite il metodo setListener. Questa flessibilità permette agli sviluppatori di adattare il comportamento di PathHandler a scenari d'uso specifici, implementando logiche di gestione personalizzate in base alle necessità dell'applicazione. Modificando il listener in PathHandler il listener che viene chiamato da Watcher rimane ugualmente corretto.

3.4 Listener

Il Listener integrato nel PathHandler è importante per collegare gli eventi rilevati dal Watcher alla logica di gestione del Bucket. Questa interfaccia funzionale, chiamata tipicamente FileChangeListener, definisce un metodo fileChange che elabora gli eventi sui file e determina quante "unità di acqua" aggiungere al bucket in base all'attività osservata nei file system monitorati.

Il metodo fileChange accetta tre insiemi (Set) come parametri:

1. deleted: l'insieme dei percorsi dei file che sono stati eliminati.
2. created: l'insieme dei percorsi dei file che sono stati creati.
3. modified: l'insieme dei percorsi dei file che sono stati modificati.

Il metodo fileChange analizza questi set e calcola un valore intero (long) che rappresenta la quantità di "acqua" da aggiungere al bucket. Il calcolo può includere metriche come l'entropia dei file modificati o creati, non considerando i file eliminati. La logica specifica può variare a seconda delle implementazioni del listener e delle esigenze specifiche dell'applicazione, non sono imposti vincoli ulteriori.

Quando il Watcher rileva cambiamenti nelle cartelle che sta analizzando invoca il fileChange e utilizza il valore restituito per aggiornare il Bucket associato nel PathHandler. Se il risultato del fileChange implica un aumento dell'acqua nel bucket, il bucket esegue i suoi controlli interni per determinare se ciò porta a un trabocco. In caso di trabocco, il metodo overflow del bucket viene chiamato per gestire la situazione, altrimenti il bucket continua a operare normalmente.

La capacità del Listener di trasformare eventi di file system in "quantità di acqua" per il bucket permette al sistema di rispondere dinamicamente a potenziali minacce o cambiamenti critici. Questo meccanismo assicura che il sistema non solo monitori i file, ma reagisca anche in maniera proporzionata all'importanza o alla gravità delle modifiche rilevate, sostenendo così l'efficacia complessiva del sistema di sicurezza o di monitoraggio implementato.

3.5 EntropyUtils

Per analizzare un file e controllare se è "sano" o è stato cifrato da un ransomware in questa implementazione viene usata la metodologia descritta nel paper di Simon R. Davies et al.[2].

La classe EntropyUtils è una classe che contiene solo metodi statici che servono a semplificare operazioni comuni basate sull'analisi dell'entropia di file, in particolare contiene lo funzioni che, dato il percorso di un file, permettono di calcolare un "punteggio" di quanto è probabile che un file sia stato cifrato secondo questa metodologia.

Questa classe inoltre genera ad ogni avvio del programma una sequenza di 40 byte casuali, con annesse entropie dei primi 8-16-...40, da tenere come "base" per la metodologia presa dal paper.

La logica è la seguente:

1. Dato il percorso del file, innanzitutto si leggono i primi 40 byte e si prendono in considerazione solo quelli.
2. Viene calcolata l'entropia di Shannon per le sottoliste di questa sequenza di byte di grandezze multiple di 8: i primi 8, i primi 16, ... fino a calcolarla per l'intero array. Si ottiene così una serie di punti con ascissa il numero di byte analizzati e ordinata l'entropia.
3. Si calcola l'area tra la curva ottenuta dai punti 'base' e i punti di questi byte: si può semplicemente fare la differenza tra i due grafici ed applicare la 'regola del trapezio' al grafico ottenuto, data la natura del grafico questo è un calcolo accurato e non una approssimazione come nel caso di una curva data da una funzione.
4. L'area ottenuta è il punteggio del file, più questo è elevato, più è alta la probabilità che il file sia cifrato.

3.5.1 Analisi preliminare

Sono state analizzate delle collezioni di file presi dal dataset NapierOne[3] per verificare l'attendibilità di questa metodologia ed ottenere una 'sicura area minima'. La versione delle collezioni usate è la 'tiny' contenente 100 file. I valori sono stati calcolati 10 volte cambiando i 40 byte di base, generandoli casualmente ogni volta.

Si può ritenere 16 un'area abbastanza sicura in quanto non solo è un'occorrenza molto rara che un'area sia inferiore a 16, ma l'unico tipo

Tabella 3.1: Plain Text, Immagini, PDF

	TXT	CSV	JSON	PNG	JPG	BMP	PDF
Max	139	38	53	26	31	77	36
Min	9	14	10	8	8	56	21
Mean	26	24	25	25	29	66	27
Median	22	24	25	26	30	65	25

Tabella 3.2: Microsoft Office, OpenDocument Spreadsheet

	DOC	DOCX	PPT	PPTX	XLS	XLSX	ODS
Max	127	66	127	64	127	61	32
Min	126	40	126	55	126	55	28
Mean	126	56	126	58	126	57	29
Median	127	58	127	58	127	57	30

di file in cui capita frequentemente sono i file txt, per cui si possono prendere precauzioni nei Listener.

Sono stati eseguiti dei test durante un utilizzo normale di un computer ed è risultato che la cartella APPDATA contiene e modifica spesso file la cui area calcolata è molto bassa. Questo è dovuto al fatto che non solo contiene file di configurazione dei programmi, ma è anche la cartella dove le applicazioni possono mettere dei file temporanei o utilizzarla come cache, questo ha l'effetto che è comune ci vengano generati file cifrati o che lo sembrano in quanto usati solo da una specifica esecuzione di una specifica applicazione, lanciando quindi molti falsi positivi su si usano bucket non calibrati attentamente per questa cartella e per specifici casi d'uso.

Nonostante alcuni ransomware bersaglino principalment questa cartella nel prossimo capitolo essa sarà esclusa dalle rilevazioni e ransomware che bersagliano non saranno utilizzati.

3.6 App.java

Nel file app.java, il metodo main orchesta il setup e la gestione del monitoraggio delle directory critiche del sistema, utilizzando la struttura e le funzionalità che abbiamo discusso.

Capitolo 4

Test

Per verificare l'efficacia dell'implementazione proposta, sono stati condotti due tipi di test: test durante un uso normale e test durante un attacco ransonware, eseguiti mediante SAFARI, una piattaforma di studio dei ransomware chiamata. I due tipi di test servono per studiare la presenza di falsi positivi (quelli corrispondenti all'uso normale del terminale) e di falsi negativi (quelli corrispondenti a un attacco che non viene rilevato).

4.1 Uso normale

Per uso normale si intendono i seguenti scenari:

- Copia di file sani su cartelle osservate: questo serve a simulare l'inserimento di nuovi dati sani in un server.
- Visualizzazione video su browser: questo serve a simulare l'utilizzo leggero da parte di un utente standard.
- Uso di videogiochi: questo serve a simulare un utilizzo più pesante rispetto alla semplice visualizzazione di video, sempre da parte di utenti standard.

Ognuno di questi test è stato eseguito due su una macchina windows per un'ora, ed in tutti i casi non sono stati osservati falsi positivi.

4.2 SAFARI

Per provare questa implementazione è stato possibile accedere ad un cluster di elaboratori su cui poter creare e distruggere macchine virtuali (VM).

Mi è stato permesso accedere a queste macchine per potere valutare l'efficacia di questo sistema, eseguendo VM che sono state rese bersaglio di specifici ransomware, i quali tratteremo in seguito. Per ogni scenario di attacco, è stata configurata una VM su cui ho installato il sistema operativo Windows 10, partendo da un template in cui è caricato anche un profilo utente generico, che ha lo scopo di simularne l'utilizzo con una serie di documenti ed immagini. Una volta avviata la VM, ho trasferito il software di individuazione e il ransomware selezionato, staccato tutte le schede di rete e avviato il ransomware. Dopo avere aspettato 300 secondi (5 minuti) che la cifratura avesse inizio e fosse ad un buon punto, ho spento la VM e analizzato il disco da un'altra VM con il sistema operativo Ubuntu 24.02, per questioni di sicurezza ed evitare l'infezione di questa macchina, per prelevare le informazioni sul tempo impiegato per la rilevazione. Ho anche controllato che il ransomware fosse partito, controllando che il disco risultasse cifrato, ed in caso contrario ho fatto ripartire il test ignorando il risultato. Il processo è stato ripetuto cinque volte per ogni ransomware, per cinque diversi tipi di ransomware. Di seguito verrà dedicato un paragrafo a ciascun ransomware, fornendo una descrizione dettagliata delle sue caratteristiche.

Nel seguente studio, verrà applicato il metodo di analisi precedentemente discusso, il quale prevede un'indagine dettagliata delle interazioni dei ransomware con specifiche directory all'interno di un sistema informatico. Per facilitare la comprensione, è utile riprendere il concetto delle "cartelle monitorate", già parzialmente introdotto nei file forniti in precedenza. In questo contesto, si presta particolare attenzione alle cartelle contenenti Immagini, Documenti e Download presenti sul Desktop. L'analisi si concentra specificamente sull'attività dei ransomware WannaCry

e Annabelle rispetto a queste directory.

Diversamente, i ransomware Ryuk e Phobos mostrano un comportamento meno prevedibile: non cifrano immediatamente i file presenti nelle cartelle standard dell'utente, ma iniziano il loro processo di cifratura dalla radice del disco, procedendo poi in maniera apparentemente casuale. Questo può comportare ritardi significativi nel tempo di rilevazione, specialmente se le directory inizialmente bersagliate non sono quelle primariamente monitorate.

Per Ryuk e Phobos, dunque, si è scelto di estendere l'osservazione a tutte le cartelle dell'utente, escludendo specificamente la cartella AppData. Quest'ultima è stata omessa a causa della sua propensione a generare falsi positivi, come evidenziato precedentemente.

Per quanto riguarda l'analisi temporale, è essenziale considerare che i tempi di cifratura possono variare notevolmente se il ransomware inizia l'attività in sistemi con più utenti. Ad esempio, se il processo di cifratura parte da un utente diverso da quello monitorato, ci potrebbero volere fino a cinque minuti prima che le modifiche siano rilevabili nel contesto dell'utente in esame. Pertanto, questo studio si focalizza sulle modifiche rilevate nelle cartelle specificate di un singolo utente, ignorando le attività che potrebbero verificarsi in altre sessioni utente.

4.2.1 WannaCry

WannaCry è un worm ransomware che nel maggio del 2017 si è diffuso rapidamente attraverso diverse reti informatiche. Una volta infettato un computer con sistema operativo Windows, il malware cripta i file presenti sull'hard disk, rendendoli inaccessibili agli utenti, e successivamente richiede un pagamento in bitcoin per la loro decrittazione.

La diffusione iniziale di WannaCry si è distinta per diversi motivi: ha colpito sistemi importanti e di alto profilo, inclusi molti appartenenti al Servizio Sanitario Nazionale britannico; ha sfruttato una vulnerabilità di Windows che si sospetta fosse stata scoperta inizialmente dalla National

Security Agency degli Stati Uniti[4]; ed è stato associato, secondo analisi di Symantec e altri ricercatori di sicurezza, al Gruppo Lazarus, un'organizzazione di cybercriminalità che potrebbe avere collegamenti con il governo nordcoreano

Il ransomware WannaCry opera attraverso un eseguibile che funziona in modo piuttosto diretto e non è considerato particolarmente complesso o innovativo: giunge sul computer infetto sotto forma di dropper, un programma autonomo che estrae altri componenti applicativi integrati al suo interno. Questi componenti includono:

- Un'applicazione per la cifratura e decifratura dei dati.
- File contenenti le chiavi di cifratura.
- Una copia di Tor, utilizzata per le comunicazioni di comando e controllo con il gruppo dietro il ransomware.

Sebbene il codice sorgente originale di WannaCry non sia stato rinvenuto né reso disponibile agli studiosi, è relativamente semplice esaminare l'esecuzione del binario. Una volta attivato, WannaCry tenta di connettersi a un URL predefinito: questo funge da interruttore di spegnimento. Se il ransomware riesce a connettersi a tale URL, si disattiva; in caso contrario, procede alla ricerca e alla cifratura di file in numerosi formati importanti, che vanno dai file di Microsoft Office a MP3 e MKV, rendendoli inaccessibili all'utente. Infine, mostra un avviso di riscatto, richiedendo una somma in Bitcoin, generalmente intorno ai 300 dollari, per decrittare i file.

WannaCry si propaga sfruttando una vulnerabilità nell'implementazione del protocollo Server Message Block (SMB) di Microsoft Windows.

Il protocollo SMB facilita la comunicazione tra i nodi di una rete e una versione non aggiornata di tale implementazione può essere indotta, tramite pacchetti appositamente costruiti, ad eseguire codice arbitrari. Questo tipo di attacco è noto come EternalBlue.

L'interesse maggiore non risiede tanto nel ransomware WannaCry in sé, quanto nel modo in cui si è diffuso attraverso EternalBlue.

Si ritiene che l'Agenzia per la Sicurezza Nazionale degli Stati Uniti (NSA) avesse scoperto questa vulnerabilità e, anziché condividerla con la comunità della sicurezza informatica, avesse sviluppato il codice EternalBlue per sfruttarla[4]. Questo exploit fu poi rubato da un gruppo di hacker noto come Shadow Brokers, che lo pubblicarono in modo oscurato in un post di natura apparentemente politica su Medium ad aprile 2017. Microsoft aveva scoperto la vulnerabilità un mese prima e aveva rilasciato una patch, ma molti sistemi rimasero non aggiornati e vulnerabili, permettendo a WannaCry di diffondersi rapidamente a partire dal 12 maggio. In seguito all'epidemia, Microsoft criticò il governo degli Stati Uniti per non aver condiviso prima la conoscenza della vulnerabilità.

Il "kill switch" di WannaCry è una funzionalità che obbliga l'eseguibile a tentare l'accesso a un URL lungo e privo di senso prima di iniziare il processo di cifratura.
Tale URL è "iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com".

In modo controintuitivo, WannaCry procede con la sua missione di ransomware solo se fallisce il tentativo di connessione a tale dominio; se riesce a connettersi, si auto-disattiva. Il motivo di questa funzionalità non è del tutto chiaro. Inizialmente, alcuni ricercatori credevano fosse un modo per i creatori del malware di interrompere l'attacco. Tuttavia, Marcus Hutchins, il ricercatore britannico di sicurezza che scoprì il tentativo di WannaCry di contattare questo URL, ritiene che fosse destinato a rendere più difficile l'analisi del codice.

Molti ricercatori eseguono il malware in un ambiente "sandbox", all'interno del quale qualsiasi URL o indirizzo IP sembra raggiungibile; inserendo codificatamente in WannaCry il tentativo di contattare un URL insensato che non si prevedeva esistesse, i suoi creatori speravano di impedire che il malware completasse le sue operazioni sotto gli occhi dei ricercatori.

Hutchins non solo scoprì l'URL codificato, ma comprò il dominio e vi caricò un sito. Molte istanze di WannaCry, di conseguenza, non riusciro-

no mai a cifrare i computer che avevano infettato, contribuendo a limitare, sebbene non a fermare, la diffusione del malware. Poco dopo essere stato celebrato come eroe per il suo ruolo nell'arresto di WannaCry, Hutchins fu arrestato per aver collaborato allo sviluppo di altro malware nel 2014. Alla fine si dichiarò colpevole delle accuse correlate, e il giudice del caso decise di non imporgli ulteriore detenzione oltre quella preventiva, riconoscendo che aveva "cambiato vita".

Il ransomware WannaCry può essere prevenuto scaricando la patch appropriata per la propria versione di Windows da Microsoft, e il modo più semplice per farlo è aggiornare il sistema operativo alla versione più recente.

Ironia della sorte, la patch necessaria era disponibile prima dell'attacco: il Bollettino di Sicurezza Microsoft MS17-010, rilasciato il 14 marzo 2017, aggiornava l'implementazione di Windows del protocollo SMB per prevenire l'infezione tramite EternalBlue. Nonostante Microsoft avesse classificato la patch come critica, molti sistemi non erano aggiornati a maggio 2017, quando WannaCry iniziò a diffondersi rapidamente. Per i sistemi non aggiornati e infettati, l'unica soluzione è ripristinare i file da un backup sicuro, questo dovrebbe insegnare l'importanza di effettuare regolarmente backup dei propri dati.

Anche se coloro che monitorarono i portafogli Bitcoin indicati nel messaggio di estorsione osservarono che alcune persone stavano pagando il riscatto, c'è poca evidenza che questi abbiano realmente recuperato l'accesso ai loro file.

WannaCry può essere rilevato esaminando attentamente i log di sistema e il traffico di rete, inoltre WannaCry non si attiva se riesce a contattare l'URL del "kill switch", può di conseguenza rimanere inattivo sull'infrastruttura senza necessariamente criptare i file. Se si possiedono macchine Windows non aggiornate, è consigliabile cercare di individuarlo prima che un cambiamento di circostanze lo attivi. SolarWinds offriva una buona guida su come utilizzare i log del server per rilevare le attività di

WannaCry: consigliano di monitorare la creazione di file, specificamente per la cifratura di file con l'estensione documentale propria di WannaCry, e di prestare attenzione al traffico in uscita per le porte SMBv1 TCP 445 e 139, così come le query DNS per il dominio del kill switch.

Il 12 maggio 2017, WannaCry si è diffuso rapidamente in internet, sfruttando EternalBlue, ma un post iniziale del blog di Symantec sulle origini di WannaCry ha anche rivelato alcune informazioni importanti e poco note su come il malware avesse iniziato a circolare mesi prima di diventare ubiquo. Symantec crede che questi prototipi siano stati inviati dallo stesso gruppo che ha lanciato l'attacco sfruttando EternalBlue, e notò "sostanziali somiglianze negli strumenti, nelle tecniche e nell'infrastruttura utilizzati dagli attaccanti" tra le versioni di WannaCry e attachi effettuati dal Gruppo Lazarus precedentemente, il che ha permesso di attribuire l'attacco ai nordcoreani[5]. Tuttavia, queste versioni precedenti utilizzavano credenziali rubate per lanciare attacchi mirati piuttosto che EternalBlue, il che significava che la sua diffusione era inferiore.

4.2.2 ANNABELLE

Il ransomware ANNABELLE, una volta eseguito, si configura per avviarsi automaticamente ad ogni riavvio del sistema. Inoltre, termina una serie di applicazioni e processi in esecuzione, come Google Chrome, Gestione Attività e MSConfig, modificando anche le Opzioni di Esecuzione dei File Immagine (IFEO) per impedire agli utenti di eseguire varie applicazioni, tra cui browser web e editor di testo.

ANNABELLE è progettato anche per modificare il MBR (Master Boot Record) del sistema, mostrando un messaggio che attribuisce il credito ai cybercriminali responsabili del suo sviluppo ad ogni riavvio del sistema. Ulteriori ricerche indicano che ANNABELLE tenta di iniettarsi anche in dispositivi rimovibili, come le chiavette USB.

Dopo aver riavviato il sistema ANNABELLE blocca lo schermo del computer e visualizza un messaggio che richiede un riscatto. Il mes-

saggio informa le vittime della cifratura e afferma che la decrittazione richiede chiavi generate unicamente per ciascuna vittima. Pertanto, per ricevere una chiave, ogni vittima deve pagare un riscatto di 0,1 Bitcoin (attualmente pari a circa 8500 dollari). Dopo il pagamento, gli utenti dovrebbero teoricamente essere in grado di decifrare i loro dati.

Fortunatamente, ANNABELLE utilizza una chiave "hard-coded", ossia la stessa chiave per cifrare i file di tutte le vittime. Tale chiave è "wHYecVx64uX2zjVedeTeyRLN", senza virgolette[6]. Di conseguenza, non è necessario pagare alcun riscatto. Il ricercatore di sicurezza nel campo dei malware, Michael Gillespie, ha rilasciato gratuitamente uno strumento di decrittazione capace di ripristinare i file cifrati da ANNABELLE sfruttando la chiave sopra menzionata[7].

Sebbene ANNABELLE sia più complesso di un normale virus di tipo ransomware (come menzionato sopra, corrompe anche il sistema, non limitandosi a cifrare i dati), condivide molte somiglianze con altri ransomware come Cypher, Russenger, THANATOS e decine di altre infezioni. Questi virus, sebbene sviluppati da cybercriminali differenti, mostrano tutti un comportamento identico: cifrano i dati e chiedono un riscatto. Le principali differenze tra questi malware riguardano solitamente l'ammontare del riscatto e il tipo di crittografia utilizzata. Purtroppo, la maggior parte di questi virus impiega algoritmi come RSA e AES, che generano chiavi di decrittazione uniche. Pertanto, a meno che il malware non sia ancora in fase di sviluppo o presenti certi bug o difetti (la chiave è hard-coded, memorizzata localmente o simili), la decifratura dei file senza il coinvolgimento degli sviluppatori (sconsigliato contattarli) è impossibile.

I virus di tipo ransomware sottolineano l'importanza di mantenere regolari backup dei dati, tuttavia, è essenziale che i file di backup siano conservati su un server remoto o su dispositivi di archiviazione esterni non collegati. In caso contrario, il malware cifrerà anche questi.

Per diffondere ransomware, i criminali informatici impiegano spesso email spam, fonti di download di software non ufficiali, strumenti di

aggiornamento software fasulli e trojan. In alcuni casi, le email spam contengono allegati dannosi (ad esempio, documenti MS Office, file JavaScript, ecc.) che, una volta aperti, scaricano e installano il malware in modo occculto. Le fonti di download di software di terze parti (siti web di download di freeware, siti di hosting file gratuiti, torrent, eMule, e simili) presentano virus mascherati da software legittimo. Di conseguenza, gli utenti vengono indotti a scaricare e installare malware.

Gli aggiornatori di software fasulli infettano il sistema sfruttando bug o difetti del software non aggiornato, oppure scaricando e installando malware anziché gli aggiornamenti veri e propri. I trojan sono i più semplici: aprono "porte d'accesso" che permettono al malware di infiltrarsi nel sistema. Fondamentalmente, le principali cause delle infezioni informatiche sono la scarsa conoscenza e un comportamento negligente. La chiave per la sicurezza informatica è la cautela.

Pertanto è essenziale prestare attenzione durante la navigazione su Internet e non aprire mai file ricevuti da indirizzi email sospetti o irriconoscibili. Queste email dovrebbero essere eliminate immediatamente, senza essere lette. È inoltre consigliato scaricare le applicazioni esclusivamente da fonti ufficiali, utilizzando un link di download diretto. I downloader e gli installer di terze parti sono spesso utilizzati per diffondere applicazioni maligne. Di conseguenza, questi strumenti non dovrebbero essere utilizzati. Inoltre, è importante mantenere le applicazioni installate aggiornate e utilizzare una suite legittima di anti-virus/anti-spyware.

Tuttavia, è fondamentale ricordare che i criminali diffondono ransomware anche tramite falsi aggiornatori. Pertanto, le app dovrebbero essere aggiornate utilizzando funzioni implementate o strumenti forniti dallo sviluppatore ufficiale, ed è fondamentale tenere aggiornato il proprio antivirus.

4.2.3 Phobos

Phobos è un ransomware che si è guadagnato una reputazione inquietante nel settore della sicurezza informatica. Emergendo alla fine del 2018, ha rapidamente mostrato le sue capacità nefaste, distinguendosi per la sua natura insidiosa e pericolosa. La sua diffusione è stata accelerata da campagne di phishing attraverso email dannose e da tecniche di compromissione del protocollo RDP (Remote Desktop Protocol), rendendolo un strumento lucrativo nel mondo del dark web, dove è commercializzato seguendo il modello ransomware-as-a-service.

All'inizio del 2019, le analisi degli esperti di sicurezza hanno evidenziato alcune caratteristiche peculiari di Phobos. Una di queste è l'uso di estensioni di file personalizzate, che includono il nome della vittima bersagliata e un indirizzo email per avviare le negoziazioni del riscatto. Questo complica notevolmente, se non rende impossibile, il recupero dei file cifrati senza la chiave di decrittazione.

Nonostante le indagini approfondite, l'identità dietro al ransomware Phobos rimane un mistero. È stato ipotizzato che possa essere opera dello stesso gruppo criminale che sta dietro a Dharma, data la somiglianza tra le loro operazioni, o che possa risultare da una collaborazione tra i membri che gestiscono sia Dharma che il ransomware Crysis. Queste, tuttavia, rimangono speculazioni.

Come già detto sopra, Phobos si propaga prevalentemente attraverso il Protocollo di Desktop Remoto (RDP), una tecnica particolarmente efficace per infiltrarsi nei server e nelle reti interne delle organizzazioni. I criminali informatici approfittano delle porte RDP poco sicure, impiegando attacchi di forza bruta per superare le difese perimetrali. In aggiunta, Phobos adotta metodi tradizionali come il phishing via email con allegati infetti, l'exploitation di vulnerabilità mediante exploit kit e gli attacchi drive-by-download da siti web compromessi.

Quando Phobos riesce a penetrare nei sistemi, gli operatori del ransomware procedono con un'accurata esplorazione della rete per assicu-

rarsi di operare senza ostacoli. Questo ransomware si caratterizza per la necessità di un'intervento umano diretto: i cybercriminali seguono da vicino le interazioni con le vittime, modulando la richiesta di riscatto in base alla gravità del danno inflitto e alle capacità finanziarie della vittima. Una volta all'interno del sistema, Phobos inizia a scansionare gli hard disk, individuando i file di maggiore importanza per poi procedere alla loro cifratura.

A differenza di altri ransomware, Phobos procede anche con l'esfiltrazione dei dati, adottando quindi la strategia della doppia estorsione: oltre a cifrare i file carica su siti esterni tipo `mega.io` i file che ritiene importanti. In questo modo si aggiunge la minaccia della distribuzione dei file attraverso aste o al pubblico: questo è particolarmente importante in caso si tratti di dati sensibili.

Phobos si concentra su file di tipo legale, finanziario, tecnico e database di password. Una azienda i cui segreti aziendali vengono venduti ai concorrenti può ricevere un duro colpo in base a cosa è stato rubato.

La cifratura dei file è realizzata impiegando algoritmi avanzati, precisamente AES-256 e RSA-2048. In fase di gestione del riscatto, Phobos modifica il nome dei file cifrati aggiungendo un'estensione che include un identificativo unico della vittima e un indirizzo email per il contatto con i criminali. Per esempio, un file denominato "bilanci.xlsx" verrebbe rinominato in "bilanci.xlsx.id[id_vittima].[email_pro_criminali]".

Al termine del processo di cifratura, Phobos crea una nota di riscatto in formato .txt, che viene accompagnata da un'immagine bitmap utilizzata come sfondo del desktop. Questa nota fornisce dettagli come l'ID della vittima, l'email per contattare i criminali, le istruzioni per il pagamento del riscatto, generalmente richiesto in Bitcoin, e offre la possibilità di decriptare gratuitamente fino a cinque file non essenziali come prova dell'efficacia del loro strumento di decrittazione. Le vittime di Phobos si trovano prevalentemente nei settori della sanità, dell'istruzione, dei servizi di emergenza e dell'amministrazione pubblica.

Anche se la maggior parte degli attacchi è stata registrata negli Stati Uniti, Phobos ha anche colpito organizzazioni nei paesi in via di sviluppo, inclusa la Nigeria, mostrando così un'ampia distribuzione geografica del suo impatto. Nel panorama attuale di minacce cibernetiche, l'attacco ransomware rappresenta una delle sfide più complesse e, purtroppo, non esiste una formula unica per garantire una protezione completa. Tuttavia, è possibile adottare una serie di misure strategiche per prevenire e mitigare i danni di queste insidie informatiche. Per ridurre la vulnerabilità agli attacchi di ransomware come Phobos, è essenziale assicurarsi che tutti i sistemi operativi e i software applicativi siano sempre aggiornati. Mantenere il software aggiornato minimizza le falle che possono essere sfruttate dai criminali informatici, e l'applicazione tempestiva delle patch di sicurezza è cruciale per la protezione contro gli exploit già noti. Inoltre, considerando che Phobos può infiltrarsi nei sistemi tramite email malevole, diventa imperativo adottare soluzioni di filtraggio email che siano in grado di intercettare e bloccare tentativi di phishing e allegati dannosi. Questo tipo di difesa è vitale per fermare il malware alla fonte prima che possa causare danni.

Un altro aspetto critico nella lotta contro Phobos è la gestione dell'accesso al Protocollo di Desktop Remoto (RDP), una delle vie preferite da questo ransomware per penetrare nei network. Ridurre la visibilità delle porte RDP e rafforzare le misure di sicurezza con soluzioni VPN robuste e autenticazione a più fattori può drasticamente diminuire il rischio di intrusioni non autorizzate. La strategia di difesa deve includere anche la realizzazione di backup frequenti e affidabili. Avere a disposizione backup recenti e ben protetti, custoditi in locazioni sicure e separate dalla rete principale, è fondamentale. Questo consente di ripristinare rapidamente i dati in caso di attacco, evitando così di cedere alle richieste di riscatto dei criminali.

Per quanto riguarda la gestione degli utenti, è vitale istruire sia i dipendenti che i clienti sulle migliori pratiche di sicurezza informatica.

Educarli sull'importanza dell'uso di autenticazione forte e sulla necessità di mantenere una vigilanza costante può significativamente rafforzare la prima linea di difesa contro gli attacchi ransomware.

Infine, l'adozione di una suite antivirus professionale ed efficace, che includa capacità di anti-phishing e isolamento delle minacce in ambienti sandbox, rappresenta un ulteriore strato di protezione. Queste tecnologie permettono di monitorare e neutralizzare attivamente potenziali minacce prima che possano compromettere la rete.

In sintesi, sebbene la minaccia di Phobos ransomware persista, la combinazione di aggiornamenti software regolari, difese email robuste, limitazioni accesso RDP, pratiche di backup solide, educazione continua sulla sicurezza e l'impiego di soluzioni antivirus avanzate costituiscono un approccio comprensivo per difendersi efficacemente. In questo campo, ogni azione tempestiva può fare la differenza, dimostrando che la sicurezza informatica inizia con l'impegno proattivo di ogni singolo utente.

4.2.4 Ryuk

Ryuk, un ransomware che ha fatto la sua prima apparizione verso la fine del 2018, è rapidamente salito alla ribalta come uno dei più pericolosi della sua categoria. A dicembre del 2018, il New York Times riportava che Tribune Publishing aveva subito un'infezione da Ryuk, che aveva interrotto le operazioni di stampa a San Diego e in Florida. L'incidente aveva coinvolto anche lo stabilimento di stampa di Los Angeles condiviso con il Wall Street Journal, influenzando la distribuzione delle edizioni del sabato dei giornali.

Ryuk, che trae origine da una precedente variante del ransomware Hermes, si è guadagnato un posto in cima alla lista degli attacchi ransomware più gravi, come sottolineato nel CrowdStrike 2020 Global Threat Report. Questo report ha evidenziato tre richieste di riscatto tra le più elevate dell'anno attribuite a Ryuk, con cifre che raggiungevano i 5,3 milioni, i 9,9 milioni e i 12,5 milioni di dollari.

Il modus operandi di Ryuk include il "big game hunting" ovvero la pratica di prendere di mira grandi aziende a livello globale.

Il nome "Ryuk" proviene dal giapponese e fa riferimento all'anime "Death Note", dove significa "dono di dio". Nonostante il nome possa sembrare inappropriato considerando le perdite finanziarie e di dati subite dalle vittime, dal punto di vista degli hacker questo potrebbe essere interpretato come un "dono" molto redditizio.

Si ritiene che Ryuk sia gestito da WIZARD SPIDER, un gruppo di cybercriminali russi, con alcune operazioni, specialmente nel settore sanitario, legate a UNC1878, un altro noto cybercriminale dell'Europa orientale.

La diffusione di Ryuk non avviene direttamente; tipicamente, gli attaccanti installano prima altri malware sulle macchine bersaglio. Una volta che Ryuk penetra in un sistema, procede disattivando 180 servizi e 40 processi che potrebbero ostacolare il suo funzionamento o che necessitano di essere fermati per facilitare l'attacco. Successivamente, inizia la fase di cifratura dei dati, mirando a file cruciali come foto, video, database e documenti utilizzando un robusto algoritmo AES-256. Le chiavi di cifratura simmetriche sono a loro volta criptate con l'algoritmo asimmetrico RSA-4096. Un algoritmo di cifratura asimmetrico usa una chiave per cifrare che si distribuisce in giro (chiamata comunemente chiave pubblica) e una per decifrare che si tiene nascosta (chiave privata); a differenza della crittografia simmetrica che usa la stessa password per entrambe le operazioni, in quella asimmetrica con la sola chiave pubblica non si può risalire a quella privata.

Ryuk dimostra anche la capacità di eseguire la cifratura su unità condivise da remoto e può riattivare i computer nella rete per estendere il suo attacco, massimizzando così il danno. Infine, gli attaccanti lasciano sul sistema compromesso delle note di riscatto, generalmente sotto forma di file testuali come RyukReadMe.txt e UNIQUE_ID_DO_NOT_REMOVE.txt, delineando le istruzioni per il pagamento del riscatto, solitamente richiesto in Bitcoin. Queste capacità conferiscono a Ryuk un'efficacia de-

vastante e una vasta portata nel panorama delle minacce informatiche globali.

Ryuk rappresenta una delle minacce ransomware più sofisticate e pervasive, sfruttando il modello di Ransomware as a Service (RaaS). Questo modello operativo consente agli sviluppatori di ransomware di offrire il loro software malevolo in uso ad altri cybercriminali, seguendo una logica simile a quella del Software as a Service (SaaS). Gli sviluppatori di Ryuk, quindi, guadagnano una percentuale sui riscatti pagati dalle vittime degli attacchi effettuati dai loro affiliati.

Un elemento particolarmente insidioso di Ryuk è il suo utilizzo del meccanismo denominato Download as a Service (DaaS), che facilita la distribuzione del ransomware. Attraverso questo servizio, gli hacker che possiedono capacità di sviluppo ma non di distribuzione possono affidarsi ad altri con competenze specifiche nella diffusione di malware per ampliare il raggio d'azione del loro ransomware.

La catena di infezione di Ryuk spesso prende avvio da tecniche di phishing, tristemente note per la loro efficacia. Secondo AdvIntel, il 91% degli attacchi ransomware inizia con email di phishing. Di conseguenza, l'addestramento degli utenti nel riconoscimento di queste minacce risulta cruciale e può drasticamente ridurre la probabilità di successo di tali attacchi.

Nel caso di Ryuk, una volta che l'utente cade nella trappola del phishing cliccando su un link malevolo, il sistema viene compromesso dal download di elementi aggiuntivi, noti come dropper. Questi possono includere malware come Trickbot, Zloader o BazarBackdoor, che facilitano l'installazione di Ryuk o di altri strumenti maligni come Cobalt Strike Beacon. Quest'ultimo serve per stabilire una comunicazione con una rete di comando e controllo (C2), essenziale per coordinare ulteriori azioni dannose all'interno della rete infettata.

Ryuk ha dimostrato anche di saper sfruttare vulnerabilità critiche, come ZeroLogon, per attaccare server Windows e diffondere ulteriormen-

te il contagio all'interno delle infrastrutture di rete delle organizzazioni colpite. Queste capacità rendono Ryuk estremamente pericoloso e testimoniano l'evoluzione continua delle strategie adottate dai cybercriminali per massimizzare l'impatto dei loro attacchi.

Le implicazioni devastanti dei ransomware possono risultare drammatiche, pertanto è cruciale prevenire l'infezione prima che essa si verifichi. Tuttavia, non è sempre possibile intercettare ogni minaccia in tempo, quindi è fondamentale che il personale addetto alla sicurezza sia sempre vigilante e pronto a identificare i segnali di avvio di un attacco per agire tempestivamente e mitigare i danni potenziali.

La complessità nel rilevare Ryuk deriva dalla sua capacità di infiltrarsi attraverso molteplici vettori di attacco.

Gli Indicatori di Compromissione (IOC) sono strumenti vitali per gli amministratori di rete e i responsabili della sicurezza, poiché permettono di identificare le prime tracce di un'infezione imminente da Ryuk. BazarLoader, noto anche come dropper o DaaS (Download as a Service), rappresenta uno dei principali metodi di ingresso per Ryuk. Questo tipo di malware ha il compito di scaricare altri malware nel sistema infetto. Tra gli IOC di BazarLoader da monitorare, si includono le attività insolite nel registro di Windows, come la creazione di attività pianificate sospette denominate "StartAd-Ad" o l'apparizione di file eseguibili con doppie estensioni, ad esempio "Report.DOC.exe".

TrickBot è un altro vettore comune per l'introduzione di Ryuk. Questo malware si manifesta tipicamente tramite file eseguibili il cui nome è costituito da una sequenza di 12 caratteri generati casualmente, come "mnfjdieks.exe", che possono essere localizzati in directory specifiche quali:

- "C:\Windows",
- "C:\Windows\SysWOW64",
- "C:\Utenti\[NomeUtente]\AppData\Roaming".

Spesso, il primo segnale di allarme per un'organizzazione è una scher-

mata di blocco del computer, che indica che Ryuk ha cifrato dati critici. In questi casi, la capacità di un'azienda di riprendersi rapidamente dipende dalla preparazione e dall'efficacia dei suoi team di risposta agli incidenti, nonché dalla disponibilità di backup dei dati realizzati offline e al sicuro da attacchi.

4.2.5 LockBit

LockBit è una variante di ransomware in circolazione dal 2019 ed è associata all'omonima organizzazione di criminali informatici. Simile a Phobos, questo ransomware viene offerto come Ransomware-as-a-Service, consentendone la licenza a vari individui o gruppi in cambio di una quota dei profitti.

Le stime suggeriscono che fino al 75% dei pagamenti del riscatto vengono ricevuti dagli aggressori associati, mentre il resto viene assegnato agli sviluppatori di LockBit. Principalmente, LockBit si diffonde tramite campagne di phishing, sfruttando vulnerabilità di sistema o sfruttando configurazioni errate nelle reti. Dopo l'infezione, ha la capacità di diffondersi autonomamente in tutta una rete aziendale.

Il sistema è dotato di un'interfaccia di negoziazione del riscatto che consente alle vittime di interagire con i criminali informatici. Le vittime ricevono indicazioni su come pagare il riscatto e, in determinate situazioni, hanno l'opportunità di negoziare l'importo del riscatto.

LockBit impiega spesso una tattica che va oltre la semplice crittografia dei dati; avverte anche che informazioni riservate o sensibili saranno rivelate se il riscatto non verrà pagato. Questo metodo di doppia estorsione accresce l'urgenza per le vittime di ottemperare alle richieste di pagamento.

Una dichiarazione congiunta di diverse agenzie governative indica che LockBit è stato il ransomware più diffuso al mondo nel 2022, con una stima del 44% di tutti gli attacchi ransomware a livello mondiale.

Da gennaio 2020 a maggio 2023, circa 1.700 incidenti ransomware negli Stati Uniti hanno coinvolto Lockbit, portando i criminali a ricevere un totale cumulativo di 91 milioni di dollari in pagamenti di riscatto. Il software noto come "LockBit" è emerso su un forum di cybercrimine in lingua russa a gennaio 2020 e il suo processo di verifica automatizzato sembra progettato per astenersi deliberatamente dal prendere di mira i sistemi locali in Russia o altre nazioni appartenenti alla Comunità degli Stati Indipendenti.

L'obiettivo primario degli attacchi del gruppo di hacker è motivato esclusivamente da motivazioni finanziarie.

4.2.6 Risultati

Poiché solo i ransomware WannaCry e ANNABELLE criptano come bersaglio principale cartelle usate comunemente per l'archiviazione di file, le cartelle osservate nei test per questi due ransomaware sono le cartelle "Desktop", "Documents" "Downloads", e "Pictures" dell'utente "IEUser".

Nei casi di Phobos, Ryuk e LockBit la cartella presa in esame è stata la cartella home dell'utente "IEUser". Questi tre ransomware non hanno preferenze sulle cartelle da criptare, quindi spesso dopo l'avvio criptano file al di fuori della cartella presa in esame, portando a tempi di rilevazione più lunghi.

Si noti che sono stati scartati eventuali test in cui il ransomware non è partito, ed in caso il test è stato rieseguito da capo.

Come si può notare il tempo impiegato per rilevare WannaCry e ANNABELLE è molto basso: a parte il primo tentativa WannaCry è stato individuato in 6 secondi o meno, mentre ANNABELLE è sempre stato trovato in meno di un secondo. La rilevazione dei ransomware Ryuk, Phobos e LockBit invece è stata piuttosto lenta: in 30-120 secondi un ransomware può creare danni ingenti al sistema.

Tabella 4.1: Tempo impiegato per rilevare la presenza di malware

	Tentativo 1	Tentativo 2	Tentativo 3	Tentativo 4	Tentativo 5
WannaCry	47s	6s	4s	3s	4s
ANNABELLE	<1s	<1s	<1s	<1s	<1s
Phobos	76s	94s	77s	83s	94s
Ryuk	32s	36s	33s	30s	28s
LockBit	108s	126s	126s	91s	98s

Una ipotesi è che la loro cifratura di file non inizi dalle cartelle osservate, in particolare se non cifrano i file in maniera sequenziale sulle cartelle, è facile che passino da file in cartelle osservate a cartelle non osservate. Questo è rilevante perché usando tempo per cifrare file non osservati i Leaky Bucket perdono acqua, rendendo meno accurata questa implementazione.

Questa però è solo una congettura, per essere certi che il problema sia questo e non un problema dell'implementazione sono necessari altri test, eventualmente su scala più ampia.

Capitolo 5

Conclusione

In questa tesi abbiamo trattato di come il Leaky Bucket possa essere un metodo nuovo per individuare in la presenza di ransomware in azione. Al momento della stesura di questo documento non sono ancora state fatti studi che sfruttino il potenziale di questo metodo per misurare la possibilità che il sistema sia sotto attacco.

L'obiettivo è stato valutare un'implementazione dell'algoritmo del "secchio che perde".

Sono stati effettuati vari test per comprovare la validità della proposta; sfortunatamente non tutti hanno dato esito di natura significative, in quanto il loro esito è stato influenzato dalla scelta arbitraria delle cartelle osservate.

L'efficacia del sistema è influenzata dal percorso di azione del ransomware, e dai file monitorati dal software che è stato predisposto a combatterlo, in quanto non è possibile individuare una azione di criptazione file indesiderata eseguita al di fuori dai file tenuti sotto osservazione.

Nei casi di WannaCry e ANNABELLE, che criptano come prima cosa i file del desktop e altre cartelle comunemente ritenute importanti per gli utenti, i risultati sono stati infatti molto soddisfacenti:

- WannaCry è stato individuato una volta in 47 secondi e in tutte le altre in 6 secondi o meno.

- ANNABELLE è stato individuato ogni volta in meno di un secondo, per la precisione nel range 0.1-03 secondi.

Se si sorveglia un numero maggiore di cartelle o addirittura tutto il disco si può presumere che la risposta di questa implementazione sarà più rapida, ma al momento non sono stati fatti test a riguardo.

Il metodo utilizzato per calcolare se un file è sano attraverso l'entropia dei primi byte formulato nel paper di Simon R. Davies et al.[2] è ortogonale al metodo di individuazione di ransomware attraverso Leaky Bucket: si può usare un algoritmo che a partire dal percorso di un file calcola se e quanta acqua aggiungere al bucket che controlla la cartella.

Questo è molto importante perché non solo permette facilmente l'utilizzo di tutta l'infrastruttura Leaky Bucket con nuovi algoritmi, ma permette anche la loro composizione. Si prendano due algoritmi che, dato in input il percorso di un file, calcolano quanta acqua aggiungere al bucket, si può facilmente creare un algoritmo che li utilizza come filtri per calcolare l'acqua da aggiungere. Un caso semplice è eseguire entrambi gli algoritmi e restituire come acqua da aggiungere un valore pari al massimo o al minimo del loro risultato. Aggiungere il massimo è utile se si vuole minimizzare la presenza di falsi negativi, ad esempio in un sistema in cui sono presenti dati estremamente importanti, ma con la possibilità di interrompere l'esecuzione senza particolari ripercussioni. Al contrario, aggiungere il minimo è utile se si vuole minimizzare la presenza di falsi positivi, ad esempio in un sistema che deve rimanere costantemente acceso e disponibile.

Si possono costruire anche casi più complessi, ad esempio se un algoritmo ha una complessità computazionale maggiore, si può considerare di eseguirlo solo se il primo algoritmo rileva un maggior rischi di corruzione (ovvero se l'acqua che aggiungerebbe supera una certa soglia).

Si può considerare una composizione in verticale l'aggiunta di algoritmi che aggiungono acqua calcolando il minimo dell'acqua aggiunta da onguno, o che eseguono solo il primo algoritmo e quelli successivi solo

in caso questo provi ad aggiungere un totale di acqua sopra una stabilita soglia: se si considera un algoritmo come un imbuto che trattiene un po' d'acqua, se il primo non fa passare nulla, anche gli altri "sotto" non faranno passare nulla, quindi non sarà aggiunta acqua.

D'altra parte si può considerare una composizione in orizzontale l'aggiunta di algoritmi che aggiungono acqua calcolando il massimo dell'acqua aggiunta da onguno, o addirittura la somma: ogni algoritmo viene eseguite e "dice la sua", senza scartare alcun risultato, si possono in questo caso immaginare gli algoritmi come tanti rubinetti che vanno a finire nel secchio, magari intersecandosi in caso di calcoli più complessi di una somma, ma che tutti contribuiscono ad aggiungere acqua.

5.1 Lavori futuri

In futuro, oltre a testare il sistema con altri ransomware e cambiando i paramentri dei Leaky Bucket o le cartelle osservate, si può introdurre una gerarchia tra i bucket: un bucket che riceve gli eventi di aggiunta di acqua da tutti gli altri può vedere le cose da un punto di vista globale invece che locale, per scongiurare attacchi da ransomware che hanno un comportamento erratico, e potrà reagire a seconda anche di quali bucket hanno aggiunto acqua al loro "capo".

L'utilizzo di una gerarchia può semplificare il controllo di sistemi con svariati dischi, ad esempio avendo tre livelli di "visione":

1. I bucket locali, come sono implementati in questo progetto.
2. Dei bucket intermedi, uno per disco o altra unità importante (ad esempio in sistemi con calcolatori multipli uno per calcolatore). Questi bucket hanno una visione d'insieme sull'unità che controllano, ma in caso queste siano più di una, non sulle altre, e ricevono dati solo dai bucket della loro unità (ad esempio dal disco che controllano).

3. Un bucket globale che riceve input dai bucket intermedi, ha una visione d'insieme su tutto il contenuto che si vuole osservare, ma a livello macroscopico: ai livelli inferiori pensano bucket di "rango" inferiore.

È importante anche riprendere il discorso iniziato nel paragrafo scorso degli algoritmi che convertono percorsi di file in acqua da aggiungere, in quanto si sposa bene con gerarchie multiple di bucket: si possono definire bucket più o meno stringenti a seconda del rango.

In questo modo si possono anche costruire livelli di importanza tra bucket dello stesso livello senza dover creare nuovi livelli gerarchici: se un bucket intermedio riceve acqua da un bucket di livello sottostante può modificare questo valore prima (o dopo) l'elaborazione dell'acqua da aggiungere in base alla tipologia di file che il bucket sottostante controlla: se viene modificato un file nella cartella con i file di boot e ne viene cambiato il proprietario, un bucket che controlla questo percorso ha più importanza di uno che controlla la cartella documenti: i documenti vengono modificati relativamente spesso, i file che controllano il boot di sistema invece tendono a rimanere relativamente fissi.

La creazione di bucket che implementano diverse tecniche contemporaneamente a seconda della cartella da osservare o, in caso di bucket di livello superiore, strategie che aggiungono più o meno acqua quando ne viene aggiunto ad un bucket di livello inferiore, permette di creare filtri che si adattano alle esigenze del sigolo caso.

Bibliografia

[1] Claude Elwood Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 1948.

[2] Simon R. Davies, Richard Macfarlane, and William J. Buchanan. Differential area analysis for ransomware attack detection within mixed file datasets. *Computers & Security*, 108:102377, 2021.

[3] Simon R. Davies, Richard Macfarlane, and William J. Buchanan. Napierone: A modern mixed file data set alternative to govdocs1. *Forensic Science International: Digital Investigation*, 40:301330, 2022.

[4] Craig Timberg Ellen Nakashima. Nsa officials worried about the day its potent hacking tool would get loose. then it did. https://arstechnica.com/information-technology/2017/04/nsa-leaking-shadow-brokers-just-dumped-its-most-damaging-release-yet, 2017.

[5] Chris Bing. Mounting evidence points to north korean group for global ransomware attack. https://cyberscoop.com/wannacry-symantec-lazarus-group, 2017.

[6] notkohlrexo. Annabelle ransomware. https://github.com/notkohlrexo/Annabelle-Ransomware/issues/2, 2020.

[7] Michael Gillespie. Stupid decryptor. https://www.bleepingcomputer.com/download/stupiddecryptor, 2020.