



**UNIVERSITÀ
DEGLI STUDI
DI BERGAMO**

Dipartimento di
Ingegneria Gestionale, dell'Informazione e della Produzione

Corso di Laurea in
Ingegneria Informatica

Classe L-8

Sistema Client-Oriented per la difesa web attiva

Integrazione di sistemi di sicurezza lo-
cale e network based per la difesa dei
client in rete

Candidato:

Davide Bonsembiante

Matricola 1086863

Thomas Paganelli

Matricola 1085805

Relatore:

Chiaramo Prof. Matthew Rossi

ANNO ACCADEMICO
2024/2025

Sommario

La crescente complessità delle minacce informatiche, unitamente alla diffusione di attacchi sempre più mirati verso i dispositivi client, ha evidenziato la necessità di un'evoluzione nei modelli di difesa informatica tradizionali. Questa tesi presenta lo sviluppo e l'implementazione di un sistema di difesa web attiva orientato al client, concepito per combinare in modo sinergico soluzioni di sicurezza locali e basate su rete. L'obiettivo principale è progettare un'architettura in grado di monitorare, rilevare e mitigare le minacce web in tempo reale, sfruttando meccanismi sia residenti sul dispositivo dell'utente sia collocati a livello di infrastruttura di rete. Dopo un'introduzione generale alle principali tipologie di minacce informatiche e ai modelli di attacco, vengono analizzati i sistemi di difesa attualmente impiegati, suddivisi tra componenti locali e network-based. La tesi descrive nel dettaglio la realizzazione di un sistema integrato costituito da:

- un **modulo client**, dotato di antivirus, quarantena file, connessione VPN e interfaccia utente intuitiva;
- un **firewall cloud di nuova generazione (NGFW)**, configurato con regole personalizzate, sistemi di rilevamento/prevenzione intrusioni, DNS Security e controllo centralizzato del traffico.

La parte finale della trattazione è dedicata alla valutazione del sistema mediante scenari di test pratici, evidenziando i benefici dell'approccio integrato in termini di efficacia e prestazioni. La tesi si conclude con una riflessione sulle limitazioni riscontrate e sulle direzioni per futuri sviluppi.

Indice

1 Minacce informatiche	1
1.1 Definizione e contesto	1
1.2 Tipologie di minacce	2
1.2.1 Vettori di attacco e tecniche malevoli	3
1.2.2 Principali categorie di minacce per obiettivo e natura	5
1.3 Origine e provenienza	9
1.4 Diffusione in rete	12
2 Sistemi di difesa	18
2.1 Concetti fondamentali dei sistemi di difesa	18
2.1.1 Definizione e scopo	18
2.1.2 Classificazione dei sistemi difensivi	18
2.2 Sistemi di difesa a livello locale	19
2.3 Sistemi di difesa a livello di rete	22
2.4 Verso un approccio integrato: sinergie tra difese locali e di rete	26
2.4.1 Vantaggi di un approccio integrato	26
2.4.2 Il caso SSL	26
3 Implementazione delle difese locali	27
3.1 Introduzione e obiettivi del componente client	27
3.2 Architettura software del componente client	29
3.2.1 Approccio asincrono e gestione dei thread	31
3.3 Integrazione del modulo di scansione antivirus locale	32
3.3.1 Scelta della tecnologia: ClamAV	32

3.3.2	Architettura e implementazione del modulo	33
3.3.3	Gestione degli aggiornamenti e delle firme malware	34
3.3.4	Implementazione della quarantena e isolamento dei file	35
	Approccio "safety-first": quarantena preventiva	36
3.4	Implementazione del modulo di monitoraggio dei download	36
3.4.1	Obiettivi e architettura	36
3.4.2	Implementazione tecnica: il WatchService di Java NIO	37
3.4.3	Gestione degli eventi e mitigazione dei rischi	38
	Gestione delle scansioni premature	38
	Prevenzione di scansioni multiple	39
3.5	Implementazione e gestione della connessione VPN	40
3.5.1	Architettura del componente VPN	40
3.5.2	Gestione del ciclo di vita della connessione	41
	Avvio della connessione	41
	Arresto della connessione	41
	Monitoraggio e raccolta delle statistiche	42
3.5.3	Gestione delle configurazioni dei peer	42
3.5.4	Integrazione avanzata con il firewall di Windows (WFP) per il kill-switch	43
	Analisi preliminare e limiti degli strumenti standard	43
	Studio del codice sorgente di WireGuard-Windows	44
	Progettazione e implementazione della soluzione	44
3.6	Sviluppo dell'interfaccia utente	46
3.6.1	Filosofia di progettazione e tecnologie	46
3.6.2	Architettura della vista principale	46
3.6.3	Gestione asincrona e aggiornamento dei dati	47
3.6.4	Progettazione a componenti riutilizzabili	48
	Schede informative per i file scansionati	49
	Pannello di gestione dei peer	50

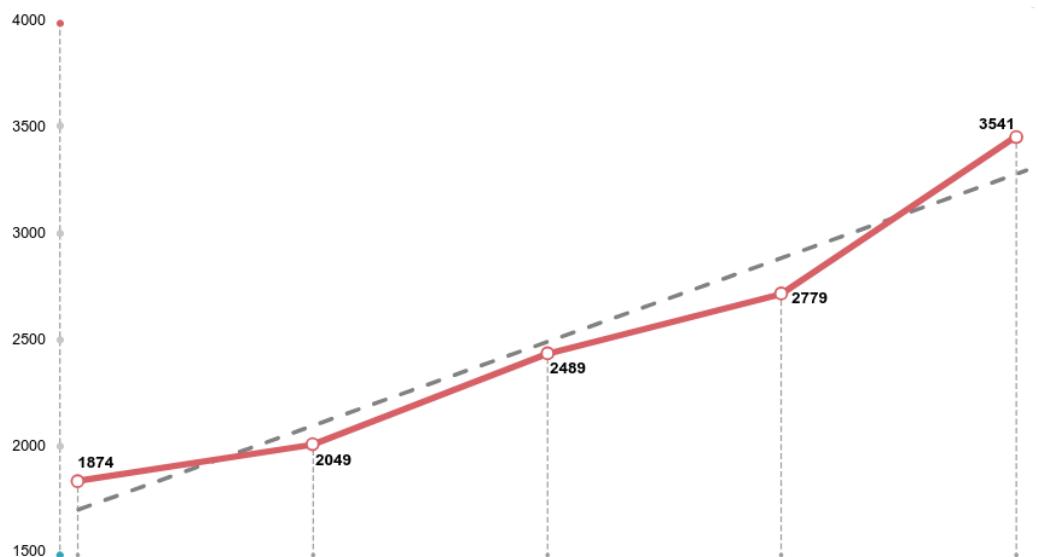
4 Implementazione delle difese in rete	53
4.1 Introduzione e obiettivi del componente NGFW cloud	53
4.2 Configurazione dell'infrastruttura di base dell'NGFW	55
4.2.1 Scelta del cloud provider e progettazione dell'infrastruttura . .	56
4.2.2 Configurazione Security List relativa alla VCN	57
4.3 Configurazione del server VPN per l'accesso client	59
4.3.1 Funzionamento e configurazione di WireGuard	60
4.4 Applicazione delle regole firewall personalizzate	61
4.4.1 Isolamento dei client connessi al NGFW	63
4.4.2 Protezione contro DNS leak e DNS-over-HTTPS (DoH)	63
4.5 Configurazione componente IPS/IDS	65
4.6 Configurazione del sistema di filtraggio DNS e DNS Security	67
5 Test, analisi prestazionale e risultati del sistema integrato	70
5.1 Metodologia di test	70
5.2 Scenari di test e risultati delle difese locali	71
5.2.1 Rilevamento basato su firme	71
5.2.2 Rilevamento euristico di malware modificato	73
5.2.3 Analisi statistica di falsi positivi e falsi negativi	76
5.2.4 Performance e tempi di scansione con ClamAV	78
5.2.5 Gestione della quarantena per file infetti e puliti	80
5.3 Scenari di test e risultati delle difese di rete	85
5.3.1 Firewall	85
5.3.2 Peer isolation	89
5.3.3 IPS/IDS	91
5.3.4 DNS filtering	94
6 Limitazioni e possibili soluzioni	98
6.1 Limitazioni del sistema	98
6.2 Possibili soluzioni	100
7 Conclusioni	102

Capitolo 1

Minacce informatiche

1.1 Definizione e contesto

Nell'era digitale, la nozione di *minaccia informatica* è diventata uno dei concetti più importanti e rilevanti, specialmente dal momento che la tecnologia e la connettività si sono integrati in ogni aspetto della nostra quotidianità, aumentando progressivamente l'esposizione ai rischi e, di conseguenza, il numero di incidenti informatici (Figura 1.1).



© Clusit - Rapporto 2025 sulla Cybersecurity

Figura 1.1: Andamento degli incidenti cyber nel periodo 2020 – 2024 [1].

Il National Institute of Standards and Technology (NIST), definisce il termine minaccia come una “*qualsiasi circostanza o evento con il potenziale di causare danni a un sistema informativo attraverso accessi non autorizzati, distruzione, divulgazione, modifica di informazioni o negazione di servizio*” [2].

Tali minacce possono avere origine sia da attori esterni — come cybercriminali o gruppi organizzati — sia da fonti interne — come dipendenti negligenti o malintenzionati. Inoltre, possono manifestarsi in modo accidentale — ad esempio a causa di errori umani o guasti tecnici — oppure in modo intenzionale — come nel caso di attacchi mirati volti a compromettere la sicurezza delle infrastrutture digitali. Nel corso del tempo, il panorama delle minacce è radicalmente cambiato da semplici attacchi — come i virus diffusi tramite supporti fisici negli anni '80 — a minacce altamente sofisticate e persistenti — come gli attacchi APT (Advanced Persistent Threats), tipici dell'ultimo decennio.

La comprensione delle minacce informatiche è fondamentale per poter progettare sistemi di difesa efficaci e capaci di tenere il passo con l'aumento costante della frequenza e della complessità degli attacchi alle infrastrutture digitali.

1.2 Tipologie di minacce

Esistono molte forme di minacce informatiche, ognuna delle quali ha caratteristiche, vettori di attacco e finalità uniche, motivo per cui è importante conoscerle per potersi preparare a individuarle e proteggersi contro di esse.

Basandoci sul rapporto *ENISA Threat Landscape 2023* [3], si propone una classificazione delle minacce distinguendo tra:

1. **Vettori di attacco e tecniche malevoli**, cioè i metodi operativi e gli strumenti utilizzati per portare a termine l'attacco.
2. **Principali categorie di minacce per obiettivo e natura**, ovvero l'impatto finale, lo scopo o la natura stessa della minaccia.

1.2.1 Vettori di attacco e tecniche malevole

In questa sezione sono descritti i metodi, gli strumenti e le strategie con cui le minacce vengono concretamente messe in atto, che possono essere combinati tra loro per creare attacchi mirati e sofisticati.

- **Social engineering:** Insieme di tecniche impiegate per sfruttare errori o comportamenti delle persone al fine di ottenere accesso non autorizzato a informazioni o servizi. È basata sulla manipolazione psicologica della vittima che spinge quest'ultima a rivelare informazioni o commettere errori. Il phishing è la tecnica più conosciuta, dove l'attaccante, principalmente per posta elettronica, simula di essere una fonte credibile e invia notifiche fraudolente con il fine di indurre l'utente a cliccare su un link o a condividere informazioni riservate.
- **Malware (malicious software):** Termine generico che descrive qualsiasi software o firmware dannoso o malevolo che esegue processi non autorizzati, in modo tale che abbia un impatto negativo sulla riservatezza, sull'integrità o sulla disponibilità del sistema.

Questa categoria comprende una vasta gamma di strumenti di attacco, tra cui i più importanti sono:

- **Ransomware:** Malware che permette agli aggressori di assumere il controllo dei dati o dei sistemi di una vittima, rendendoli inaccessibili e richiedendo un riscatto per ripristinare l'accesso a tali risorse. La definizione di ransomware è stata recentemente estesa anche alla doppia e tripla estorsione, in cui i dati vengono non solo bloccati, ma anche esfiltrati e resi pubblici. Sebbene il movente sia principalmente finanziario, gli attacchi ransomware possono anche essere effettuati per motivi politici e di sabotaggio. Questa minaccia rimane una delle più cruciali e onnipresenti nel suo genere, con numerosi incidenti di alto profilo.
- **Spyware:** Malware progettato per raccogliere informazioni sensibili su un utente o su un sistema, come credenziali di accesso, cronologia di navigazione o comunicazioni, senza il consenso dell'interessato. È spesso

utilizzato per attività di spionaggio informatico, sorveglianza o furto di identità.

- **Adware:** Software malevolo o indesiderato che genera pubblicità invadenti o reindirizza l’utente verso contenuti promozionali. Anche se meno pericoloso rispetto ad altre forme di malware, può compromettere l’esperienza d’uso e talvolta servire da vettore per infezioni più gravi.
 - **Cryptojacker:** Malware progettato per sfruttare le risorse computazionali di un dispositivo infetto — come CPU e GPU — al fine di minare criptovalute a vantaggio dell’attaccante, senza il consenso o la consapevolezza dell’utente. Questo tipo di attacco può ridurre drasticamente le prestazioni del sistema e aumentare il consumo energetico.
 - **Malware per botnet:** Software malevolo che infetta dispositivi con l’obiettivo di trasformarli in “bot” controllabili da remoto, spesso all’insaputa del proprietario. I dispositivi compromessi vengono collegati tra loro a formare una botnet, che può essere utilizzata per attacchi DDoS, spam, diffusione di malware, click fraud o tentativi automatizzati di accesso a servizi (credential stuffing).
-
- **Supply chain attacks:** Attacchi volti a compromettere i sistemi di un fornitore, partner o produttore di software per inserire codice maligno o alterare servizi e prodotti che l’utente finale si aspetta di utilizzare in modo sicuro. L’obiettivo è colpire simultaneamente sia il fornitore che il cliente, sfruttando la relazione di fiducia tra i due. Un esempio ben noto è il caso di *SolarWinds*, in cui attori statali hanno sfruttato l’aggiornamento software di Orion per ottenere accesso non autorizzato a numerosi enti governativi e aziende per mesi.
 - **Zero-day exploits:** Gli exploit sono tecniche o frammenti di codice che mirano a sfruttare vulnerabilità specifiche, talvolta chiamate falle di sicurezza, esistenti nel software, hardware o sistema operativo, per ottenere accesso non autorizzato o elevare i privilegi. Le vulnerabilità zero-day, d’altro canto, sono

falle sconosciute ai produttori o al pubblico; di conseguenza, non ci sono patch disponibili, rendendo tali exploit estremamente pericolosi.

- **Man-in-the-middle (MITM):** Attacco in cui due parti ignare comunicano tra loro mentre un utente malintenzionato intercetta e, potenzialmente, modifica i loro messaggi. Può portare alla compromissione della privacy e dell'integrità dei dati trasmessi.
- **Web attacks:** Attacchi mirati ad applicazioni web e siti web in generale. Tra i più rilevanti, ci sono SQL Injection, Cross-Site Scripting (XSS), Broken Authentication e Session Management, Cross-Site Request Forgery (CSRF). Questi possono portare a furti di dati, website defacement o accesso non autorizzato.

1.2.2 Principali categorie di minacce per obiettivo e natura

Le seguenti categorie descrivono l'impatto finale o la natura della minaccia, spesso realizzate attraverso uno o più dei vettori sopra elencati.

- **Data-related threats:** Riguardano la compromissione di dati sensibili, sia in modo intenzionale che accidentale. Si distinguono principalmente in:
 - **Data breach:** Accesso o divulgazione non autorizzata di informazioni riservate, generalmente dovuta ad attacchi informatici o violazioni di sicurezza. In questo tipo di minaccia, la compromissione è attiva e intenzionale, spesso causata da hacker o insider con scopi malevoli, come il furto o la divulgazione illegale dei dati.
 - **Data leak:** Esposizione accidentale o involontaria di dati sensibili, tipicamente causata da errori di configurazione, gestione inadeguata o dissattenzioni umane. Non deriva da un attacco esterno, ma da una perdita involontaria di riservatezza, ad esempio a seguito della diffusione accidentale di informazioni o dello smarrimento di dispositivi contenenti dati.

- **Availability threats:** Rientrano in questa categoria quegli attacchi che hanno l’obiettivo di rendere inutilizzabili o inaccessibili servizi digitali fondamentali. Le principali forme includono:

- **Denial of Service (DoS/DDoS):** Attacchi volti a rendere non disponibili i servizi o le risorse informatiche sovraccaricandoli di richieste. Gli attacchi DDoS (Distributed Denial of Service) coinvolgono più dispositivi coordinati, spesso facenti parte di una botnet, e possono colpire sistemi su larga scala, inclusi dispositivi IoT compromessi. In questo caso, la botnet rappresenta il vettore tecnico, mentre il DDoS è l’effetto finale.
- **Internet outages:** Si riferisce all’interruzione, parziale o totale, della connessione a internet o di altri servizi ad essa associati, come la messaggistica istantanea o la posta elettronica. Tali interruzioni possono verificarsi a seguito di eventi accidentali — come disastri naturali o malfunzionamenti tecnici — oppure mediante atti intenzionali — come hacking, atti di terrorismo, crimini o censura governativa. Qualunque ne sia la causa, la continuità dei servizi online è messa a rischio.

- **Foreign Information Manipulation and Interference (FIMI):** È una strategia, spesso lecita, messa in atto da attori statali o non statali che cercano in modo deliberato e organizzato di influenzare pubblici di riferimento, processi democratici, valori o istituzioni. La manipolazione dell’informazione ha natura ingannevole e può avvenire tramite la diffusione di disinformazione, la modifica dei fatti o la creazione di contenuti falsi o fuorvianti e la loro diffusione attraverso internet e i social media. L’obiettivo è minare la fiducia nelle istituzioni, polarizzare il dibattito pubblico o interferire nei processi decisionali interni di un Paese.

Nota: Sebbene FIMI (Foreign Information Manipulation and Interference) e l’ingegneria sociale condividano l’uso della manipolazione psicologica, si collocano in categorie differenti di minaccia, in quanto differiscono per:

- **Attori coinvolti:** L’ingegneria sociale è tipicamente condotta da cyber-criminali o insider, mentre FIMI coinvolge quasi sempre attori statali o strutture organizzate (es. gruppi APT).
 - **Obiettivo:** L’ingegneria sociale punta a ingannare singoli utenti per ottenere accessi, credenziali o denaro; FIMI ha lo scopo di influenzare l’opinione pubblica, alterare il dibattito politico o delegittimare istituzioni.
 - **Scala e durata:** L’ingegneria sociale è episodica e puntuale, mentre FIMI si concretizza in campagne coordinate, su scala nazionale o globale, e con effetti nel lungo periodo.
 - **Canali usati:** L’ingegneria sociale si veicola attraverso email, telefonate, SMS (es. phishing, vishing, smishing); FIMI agisce attraverso social media, siti web, media tradizionali e reti di disinformazione.
 - **Misurabilità dell’impatto:** L’ingegneria sociale ha effetti tangibili su singoli bersagli; FIMI agisce in modo sistematico, ed è valutabile tramite indicatori sociopolitici, sondaggi e metriche di engagement online.
- **Insider threats:** Identificano i rischi derivanti da utenti interni in possesso di accesso autorizzato ai sistemi o ai dati di un’azienda — come dipendenti, ex dipendenti o fornitori. Queste minacce possono rappresentare una minaccia sia in modo consapevole — ad esempio a scopo di guadagno o vendetta — sia inconsapevole — ad esempio per negligenza o a seguito di furto di credenziali.
 - **Advanced Persistent Threats (APT):** Attacchi informatici altamente sofisticati e mirati, comunemente eseguiti da entità sponsorizzate dallo stato che mirano a ottenere accesso prolungato ai sistemi o alle reti colpiti, rimanendo inosservati per periodi prolungati e adattandosi alle difese.

Tra queste, il ransomware si conferma come la minaccia principale in termini di impatto e frequenza, seguito dagli attacchi DDoS e dagli attacchi che sfruttano la manipolazione dei dati (Figura 1.2).

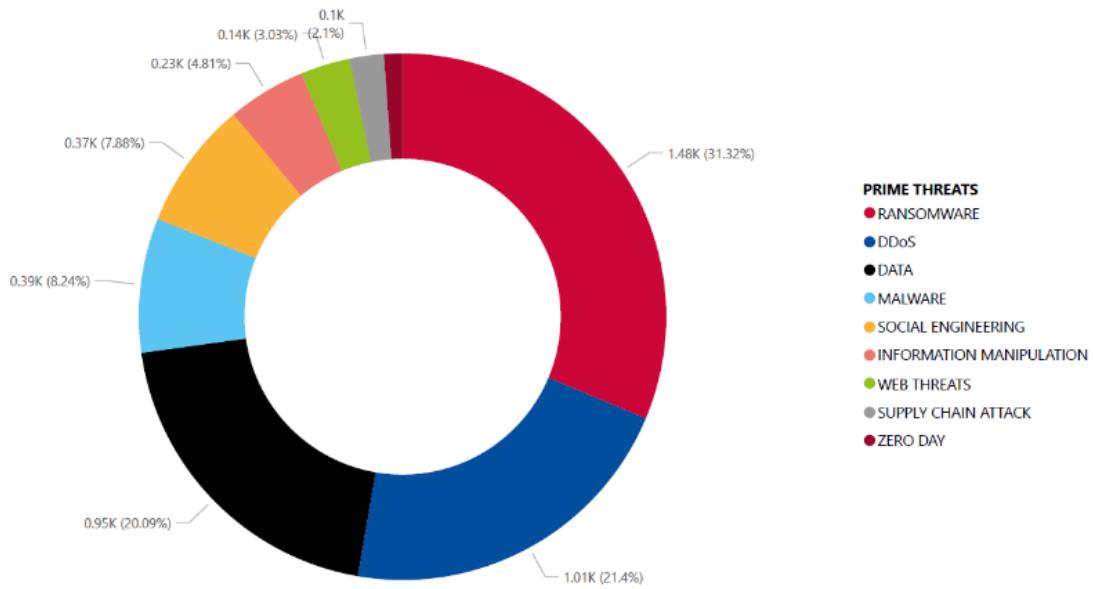


Figura 1.2: Ripartizione degli incidenti analizzati per tipo di minaccia (luglio 2022 – giugno 2023) [3].

La Figura 1.3, basata sui dati raccolti nel rapporto *Clusit 2025* [1], mostra invece la distribuzione temporale delle tipologie di minacce nel periodo 2020-2024, evidenziando la portata globale del fenomeno.

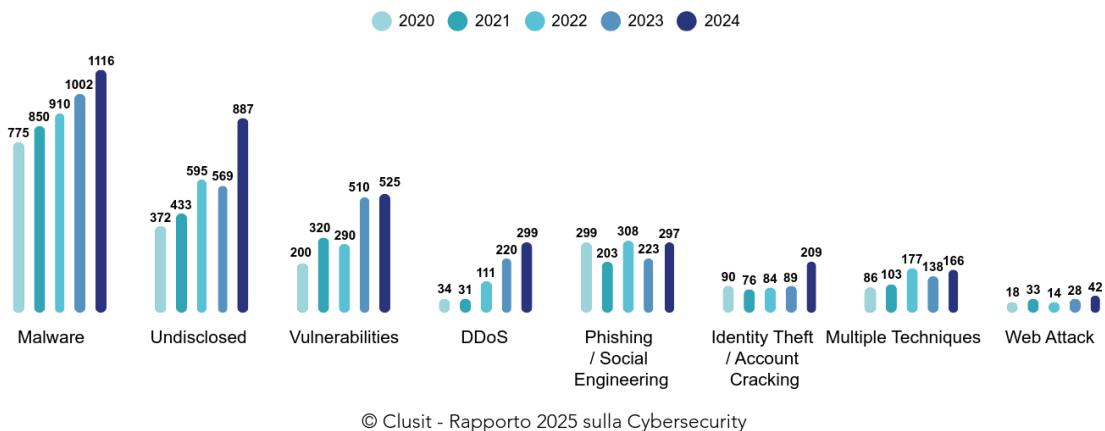


Figura 1.3: Distribuzione temporale delle tipologie di minacce nel periodo 2020-2024 [1].

Queste minacce sono dannose per gli endpoint e possono operare in modo singolo o sinergico per produrre un attacco complesso, concentrato e persistente. Inoltre, la loro diffusione è favorita dall'uso di vecchie versioni di software, impostazioni di configurazione errate, dispositivi non supportati e scarsa conoscenza degli utenti in materia di sicurezza informatica.

1.3 Origine e provenienza

Per valutare il rischio a cui sono esposti sistemi, reti e dati, è fondamentale comprendere l'origine e la provenienza delle minacce informatiche, in quanto può non solo aiutare a prevedere alcuni vettori di attacco, ma anche a personalizzare le contromisure.

Secondo il rapporto NIST [2] e il rapporto ENISA [3], una minaccia può avere origine da sorgenti esterne o interne, intenzionali o accidentali, ciascuna dotata di diversi livelli di sofisticazione, di scopo e di caratteristiche.

Le minacce possono essere classificate in base alla loro origine:

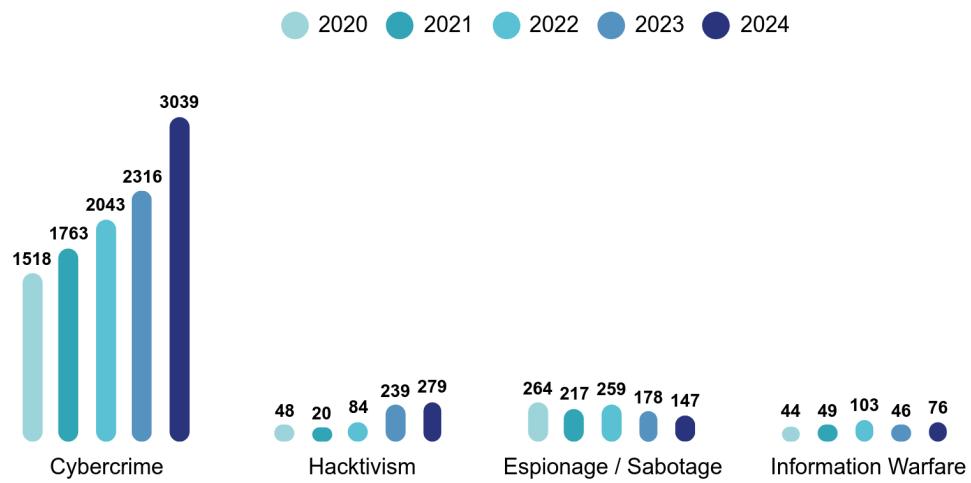
- **State-sponsored attackers (attori statali e gruppi sponsorizzati da stati)**: Agiscono con risorse e capacità elevate, spesso mirando a obiettivi strategici, militari o geopolitici. Questi gruppi, che in alcuni casi vengono definiti "APT" (*Advanced Persistent Threats*), cooperano in modo prolungato, cercando di rimanere a lungo non rilevati nelle reti o nelle infrastrutture governative. Esempi noti sono il gruppo russo APT28 (Fancy Bear), l'iraniano APT35 (Charming Kitten) e il gruppo cinese APT41 [3, 4, 5].
- **Organized cybercriminals (cybercriminali organizzati)**: Concentrati sul profitto economico, i gruppi di cybercriminali utilizzano tecniche come ransomware, phishing e frodi online per attaccare enti pubblici, ospedali e organizzazioni private. Sfruttano spesso i marketplace del dark web, acquistando e utilizzando software e servizi già pronti all'uso [6, 7].
- **Hacktivists (hacktivisti)**: Individui o gruppi che conducono attacchi per supportare posizioni politiche, etiche o sociali. Generalmente prendono di mira

i servizi web attraverso attacchi di defacement, praticando attacchi DDoS e ricorrendo talvolta a violazioni dei dati. Un esempio storico è Anonymous, attivo in diverse campagne globali.

- **Cyber terrorism (terrorismo informatico):** Alcuni gruppi terroristici hanno dimostrato la capacità di utilizzare strumenti informatici per diffondere propaganda, compiere campagne di disinformazione o tentativi di attacco ai sistemi critici. Ad esempio, L'*Islamic State Hacking Division* pubblica campagne di intimidazione e dati sensibili [5, 8].

Nota: le minacce interne e quelle di natura accidentale sono trattate nel capitolo 1.2, in quanto rappresentano specifiche modalità di attacco piuttosto che categorie riconducibili a un'origine esterna.

L'analisi delle origini degli attacchi conferma la netta predominanza dei gruppi di cybercriminali organizzati, che costituiscono la principale fonte di minaccia nel panorama attuale (Figura 1.4).



© Clusit - Rapporto 2025 sulla Cybersecurity

Figura 1.4: Distribuzione degli attaccanti dal 2020 al 2024 [1].

Le motivazioni degli attori malevoli variano in base ai loro obiettivi e, come mostrato nella Figura 1.5, la principale motivazione è il guadagno finanziario.

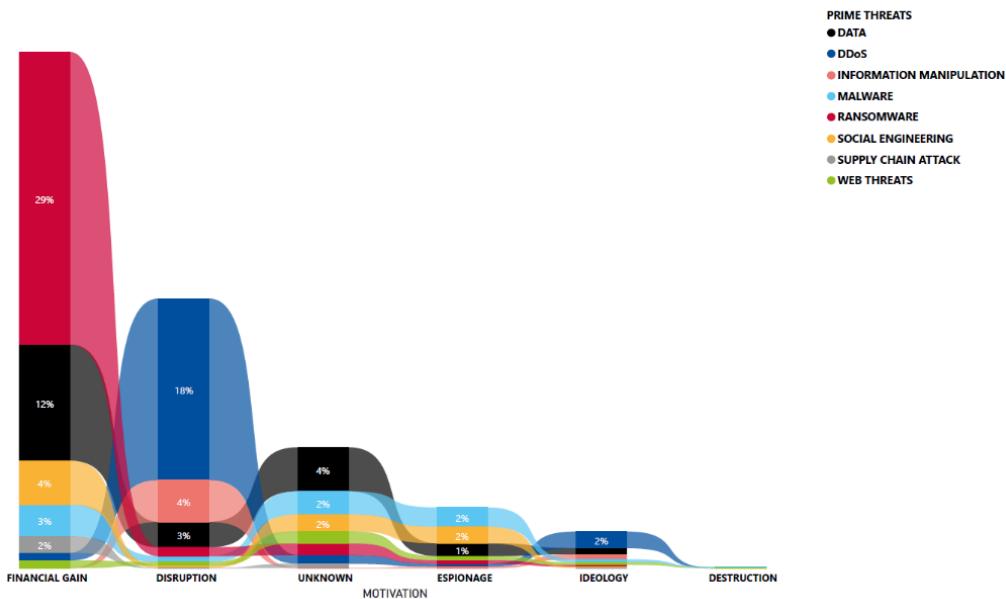


Figura 1.5: Motivazione degli attori malevoli in base alla tipologia di minaccia [3].

La componente geopolitica è oggi una caratteristica rilevante, specialmente per gli attacchi sponsorizzati dai governi. Secondo il *Cyber Operations Tracker* del Council on Foreign Relations [4], almeno 34 paesi sono stati coinvolti in operazioni offensive nel cyberspazio dal 2005 ad oggi, con Cina, Russia, Iran e Corea del Nord come principali attori. Questi paesi sono noti per spionaggio, sabotaggio e influenza, spesso condotte in modo persistente e sotto copertura.

Un elemento considerevole che conferma l'evoluzione delle minacce informatiche è la progressione storica delle tecniche di attacco. Dalla diffusione dei primi virus come il worm "Morris" (1988) [9] e il malware "ILOVEYOU" (2000) [10, 11], si è passati ad attacchi estremamente complessi come Stuxnet (2010) [12], Operation Aurora [13] e SolarWinds (2020) [14], che hanno evidenziato come la cybersicurezza sia divenuta uno strumento strategico di guerra, spionaggio e geopolitica [1, 15].

1.4 Diffusione in rete

Con l'aumento costante dei dispositivi connessi e il conseguente aumento della connettività globale, i dispositivi e le reti informatiche stesse sono sempre più un bersaglio critico; l'era della digitalizzazione, per quanto permetta una maggiore efficienza e accessibilità, espone anche i sistemi critici a minacce informatiche sempre più sofisticate e pervasive.

Per avere una stima quantitativa che metta in prospettiva il fenomeno, possiamo basarci su alcuni report istituzionali, affrontando prima il fenomeno in modo generale, e successivamente focalizzandoci sulle minacce coerenti al nostro caso di studio.

Come testimoniato da un recente studio di ENISA [3], la frequenza degli incidenti informatici che hanno interessato il Vecchio Continente è in continuo aumento (Figura 1.6), tendenza che ritroviamo anche a livello globale.

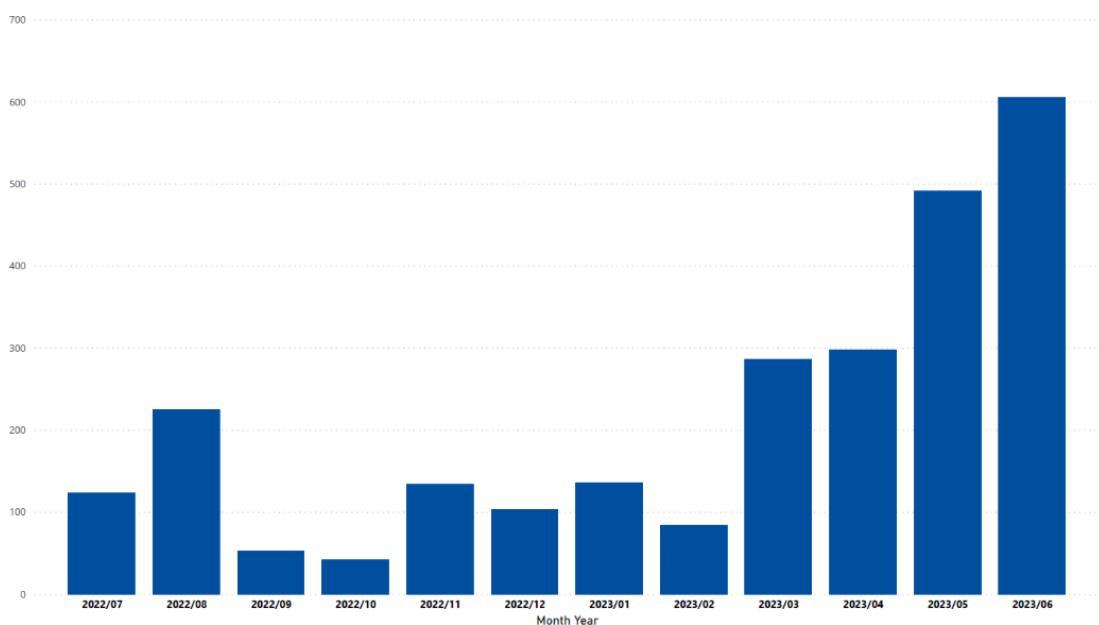
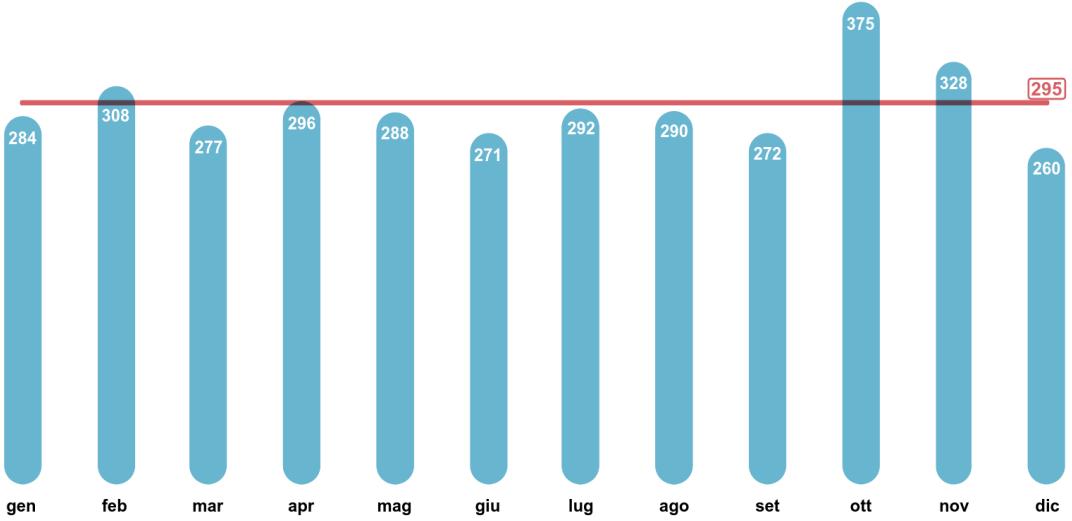


Figura 1.6: Distribuzione temporale degli incidenti di sicurezza informatica nell'Unione Europea (luglio 2022 – giugno 2023) [3].

Secondo il rapporto Clusit 2025 [1], nel solo anno passato sono stati registrati 3.541 incidenti gravi, il numero più alto mai rilevato, con una crescita del 27% rispetto ai 2.779 casi del 2023 (Figura 1.7).



© Clusit - Rapporto 2025 sulla Cybersecurity

Figura 1.7: Andamento incidenti cyber per mese 2024 [1].

In particolare, gli eventi dal 2020 al 2024 rappresentano il 56% di tutti gli incidenti analizzati dal 2011, evidenziando un aumento significativo rispetto allo scenario del 2019, con un aumento del 112%.

Secondo il DDoS Threat Report di Cloudflare [16], datato Q4 2023, la frequenza e l'intensità degli attacchi DDoS (Distributed Denial of Service) sarebbe in costante aumento, con un incremento (nel periodo di riferimento) del 117% rispetto all'anno precedente (Figura 1.8); in modo simile, anche la portata degli attacchi è aumentata, favoriti dalla continua diffusione di dispositivi IoT infetti, utilizzati per creare botnet potenti e difficili da contrastare.

Largest HTTP DDoS attacks



As seen by Cloudflare, by year

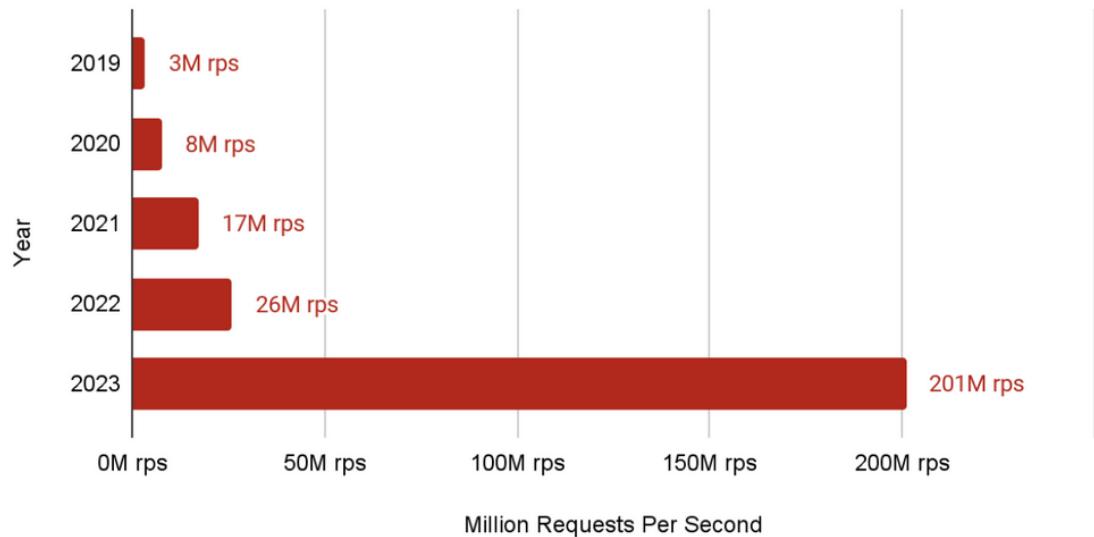


Figura 1.8: Dimensione attacchi DDoS per anno [16].

Queste tendenze si riconfermano anche nel più recente report di Cloudflare [17], datato Q1 2025, che verifica un significativo aumento degli attacchi anche sui layer di trasporto (Figura 1.9).

DDoS attacks by quarter

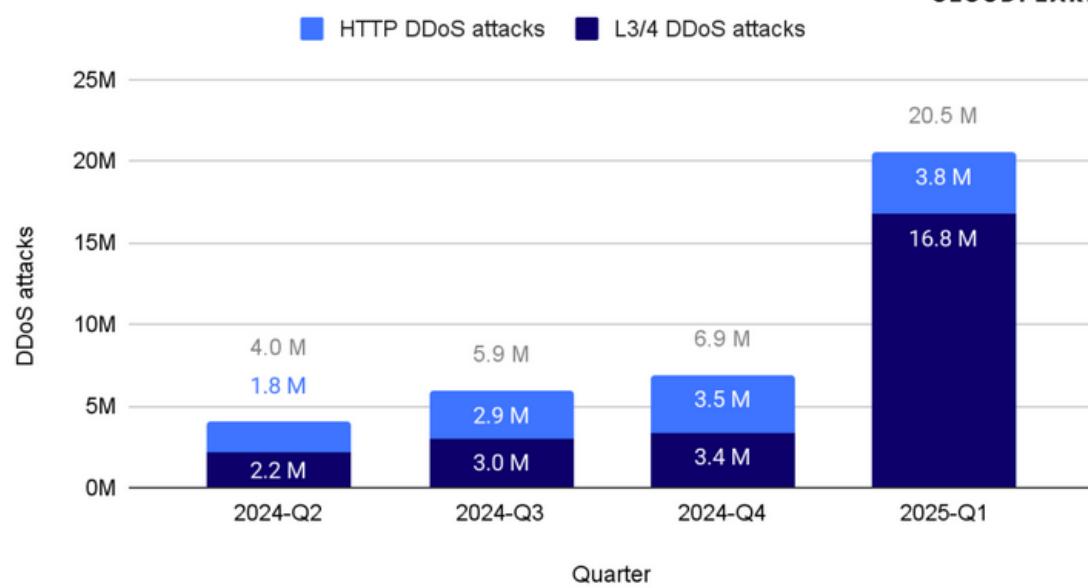


Figura 1.9: Dimensione attacchi DDoS per quadrimestre [17].

Quanto visto finora ci permette di delineare uno scenario globale, definendo l’andamento del settore Cybercrime-as-a-Service nel breve e medio periodo, permettendoci di scendere più nel dettaglio considerando tipologie di attacco più specifiche del nostro caso di studio.

Gli attacchi basati sull’ingegneria sociale continuano ad essere minacce diffuse e sempre più pericolose: grazie all’utilizzo dell’intelligenza artificiale i criminali informatici sono in grado di creare messaggi sempre più convincenti e personalizzati, aumentando la probabilità che gli utenti ne cadano vittima. Essi rappresentano il vettore di intrusione predominante in aree come l’Europa, il Medio Oriente e l’Africa (Figura 1.10).

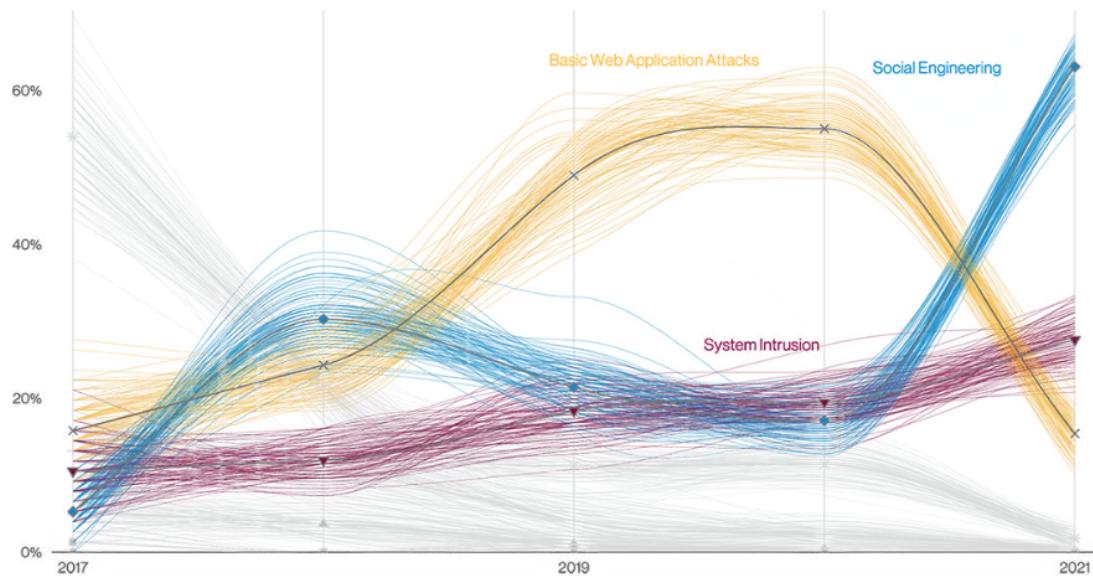


Figura 1.10: Andamento tipi di attacco in Europa, Medio Oriente e Africa [18].

Fortemente legato a questo fenomeno è l'azione dei ransomware, spesso distribuiti anche tramite campagne mirate di phishing che, secondo il Ransomware Annual Report 2024 di CyberInt [19], ha visto un aumento dell'11% rispetto all'anno precedente, con un numero di attacchi verificati che ha raggiunto i 5414 (Figura 1.11).



Figura 1.11: Andamento degli attacchi ransomware [19].

Gli incidenti che coinvolgono campagne ransomware possono derivare da malware altamente complessi, spesso classificati come Advanced Persistent Threats (APT), dotati di capacità avanzate per propagarsi autonomamente all'interno dell'infrastruttura di rete. In particolare, una singola macchina compromessa può fungere da punto d'ingresso per l'esecuzione di tecniche di lateral movement, come l'impiego di credenziali rubate o mediante exploit di vulnerabilità note su protocolli di rete come SMB (es. EternalBlue). Circa il 19% degli incidenti ransomware aziendali analizzati presenta infezioni che includono malware worm-like, ovvero capaci di replicarsi automaticamente e autonomamente tra host all'interno dello stesso segmento di rete, senza la necessità di intervento manuale da parte dell'attaccante (Figura 1.12).

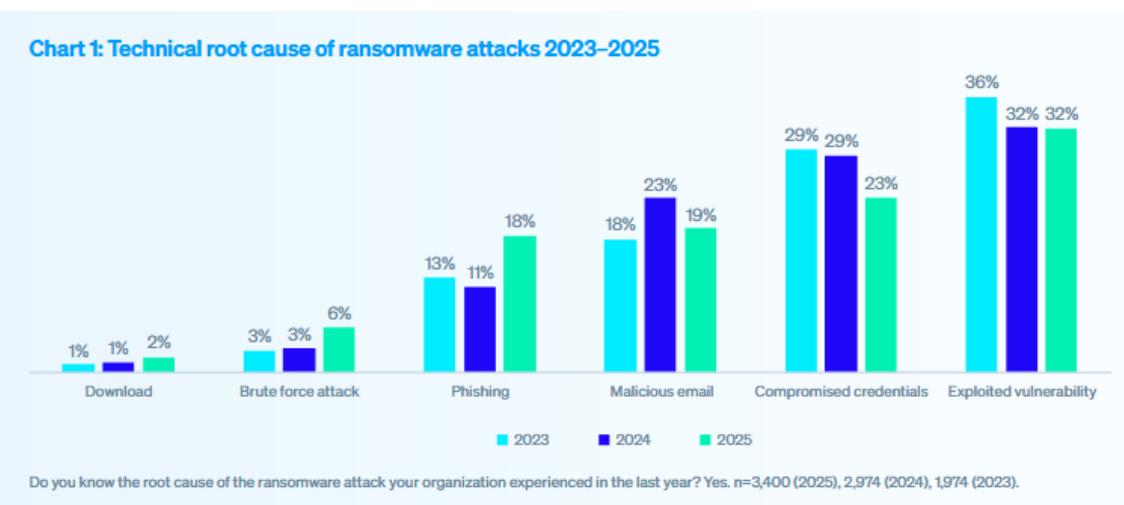


Figura 1.12: Cause principali degli attacchi che hanno fatto uso di ransomware [20].

I dati appena presentati mostrano una situazione in continua evoluzione, in cui le minacce informatiche si diversificano, adattandosi alle nuove tecnologie e ai cambiamenti nel comportamento degli utenti; con ogni probabilità, la tendenza al rialzo degli attacchi continuerà, rendendo sempre più importante l'adozione di misure di sicurezza adeguate e la formazione degli utenti per prevenirne e mitigare gli effetti.

Capitolo 2

Sistemi di difesa

2.1 Concetti fondamentali dei sistemi di difesa

2.1.1 Definizione e scopo

Un sistema di difesa è un insieme di strumenti, tecnologie, regole e pratiche pensate per proteggere reti, dispositivi, dati e applicazioni da accessi non autorizzati, furti o interruzioni del servizio. Lo scopo principale di questi sistemi è garantire la sicurezza, la riservatezza e l'integrità delle informazioni, riducendo i rischi e rispondendo agli attacchi nel modo più efficace possibile [21].

2.1.2 Classificazione dei sistemi difensivi

I sistemi difensivi si possono classificare in alcune macrocategorie, ognuna con un ruolo specifico nella protezione complessiva:

- **Sistemi di rilevamento:** Si occupano di individuare comportamenti sospetti o attività anomale all'interno del sistema; un esempio sono i sistemi IDS (Intrusion Detection System - *Fortinet* [22]), che monitorano il traffico per segnalare eventuali intrusioni, e le soluzioni antivirus, che permettono la rilevazione di eventuali malware presenti sui dispositivi. Questi sistemi possono essere potenziati da SIEM (Security Information and Event Management - *Cisco* [23]), che raccolgono e analizzano i log per identificare modelli di attacco.

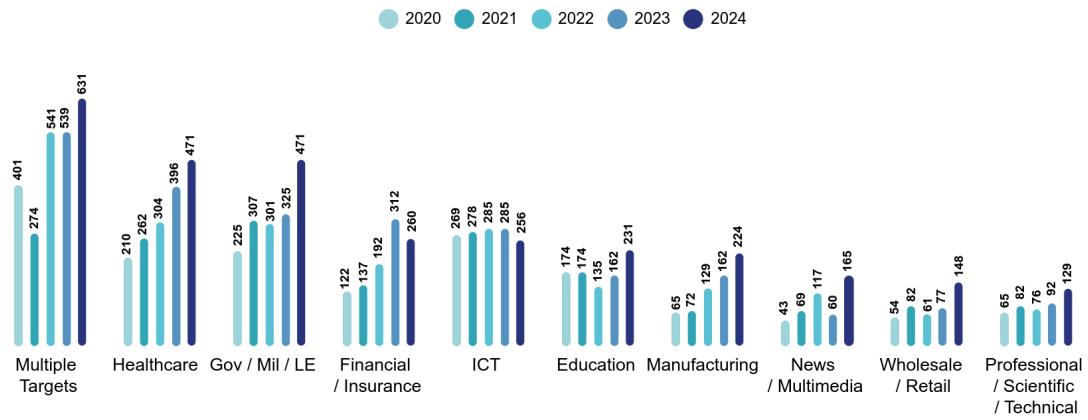
- **Sistemi di prevenzione:** Sono strumenti volti a impedire che le minacce si manifestino, agendo prima che si verifichi un attacco. Esempi tipici sono gli IPS (Intrusion Prevention Systems - *Fortinet* [24]), che filtrano il traffico in entrata e in uscita mettendo in atto azioni proattive di blocco o limitazione delle connessioni sospette; oppure i firewall (*Cisco* [25]), i quali, utilizzando pattern definiti dall'amministratore di rete, possono bloccare il traffico su determinati protocolli o porte, impedendo l'accesso a risorse non autorizzate o ritenute insicure.
- **Sistemi di risposta e contenimento:** Entrano in azione quando una minaccia viene rilevata, cercando di limitarne i danni. Quando parliamo di sistemi di risposta, oltre ai classici antivirus, ci riferiamo a strumenti che possono essere integrati con i sistemi di rilevamento e prevenzione, come i SOAR (Security Orchestration, Automation and Response - *Fortinet* [26]), che permettono l'orchestrazione della risposta alla minaccia, o i sistemi EDR (Endpoint Detection and Response - *Fortinet* [27]), che permettono la segmentazione di porzioni di rete o l'isolamento di determinati terminali per prevenire la propagazione dell'attacco.
- **Sistemi di recupero e resilienza:** Servono a ripristinare il normale funzionamento dopo un attacco, riducendone l'impatto; in questa categoria rientrano i backup e i piani di disaster recovery delle singole organizzazioni.

In questo documento ci concentreremo principalmente sui sistemi di rilevamento e sistemi di prevenzione, utilizzando applicativi pensati sia per essere implementati a livello locale, ovvero sui singoli client, sia a livello di rete, per proteggere l'intera infrastruttura.

2.2 Sistemi di difesa a livello locale

La difesa a livello locale è oggi una parte essenziale delle architetture di sicurezza moderne, data la focalizzazione sempre più importante da parte delle minacce informatiche verso i dispositivi endpoint.

A supporto di questa evidenza, la Figura 2.1 mostra la distribuzione degli attaccanti nel periodo 2020-2024, sottolineando l'urgenza di proteggere adeguatamente i clienti.



© Clusit - Rapporto 2025 sulla Cybersecurity

Figura 2.1: Distribuzione degli attaccanti 2020-2024 [1].

I sistemi di difesa a livello locale sono strumenti software installati sui dispositivi degli utenti, come computer, smartphone e tablet. La loro funzione principale è monitorare e identificare attività dannose e bloccarle prima che possano fare danno. Alcuni degli strumenti di difesa locale più diffusi sono antivirus, firewall per host, VPN, e per i dispositivi più avanzati, sistemi di rilevamento comportamentale, sandboxing, ecc.

Gli antivirus sono tradizionalmente il primo strato di protezione sugli endpoint e si distinguono in due categorie: le soluzioni commerciali e le soluzioni open-source.

Le prime – quali ad esempio Bitdefender, Norton o Kaspersky – si basano su database di firme sempre aggiornate e integrano meccanismi euristicci, machine learning e analisi comportamentali. Sono in grado di rilevare un'ampia varietà di minacce conosciute e ignote, fornire protezione in tempo reale, eseguire il sandboxing automatico, proteggere la navigazione web e monitorare le email. Tuttavia presentano degli svantaggi, come il costo e le preoccupazioni relative alla privacy, specialmente in ambito aziendale o governativo [5]. Inoltre, utilizzando soluzioni proprietarie, il loro codice non è open-source e quindi non consentono l'audit da parte di terze parti. Questo limita la trasparenza, ma al contempo può rappresentare un ostacolo per

gli utenti malevoli, in quanto non possono analizzare liberamente il funzionamento interno del software.

Le soluzioni open-source – come ClamAV o Linux Malware Detect – sono molto più trasparenti, flessibili e risultano essere particolarmente adatte quando le prestazioni in termini di leggerezza o l'integrazione con altri componenti open-source costituiscono requisiti fondamentali, pur essendo generalmente meno efficaci per il rilevamento proattivo della minaccia.

La scelta tra le soluzioni commerciali e open-source dipende dall'ambiente organizzativo. È quindi necessario valutare la quantità di risorse economiche e tecniche disponibili nelle attività aziendali, i livelli di controllo e di rischio a cui è esposto l'ambiente operativo, considerando anche l'ampiezza della superficie di attacco, con le potenziali criticità derivanti dall'evoluzione delle minacce malware più recenti [5] (Figura 2.2).

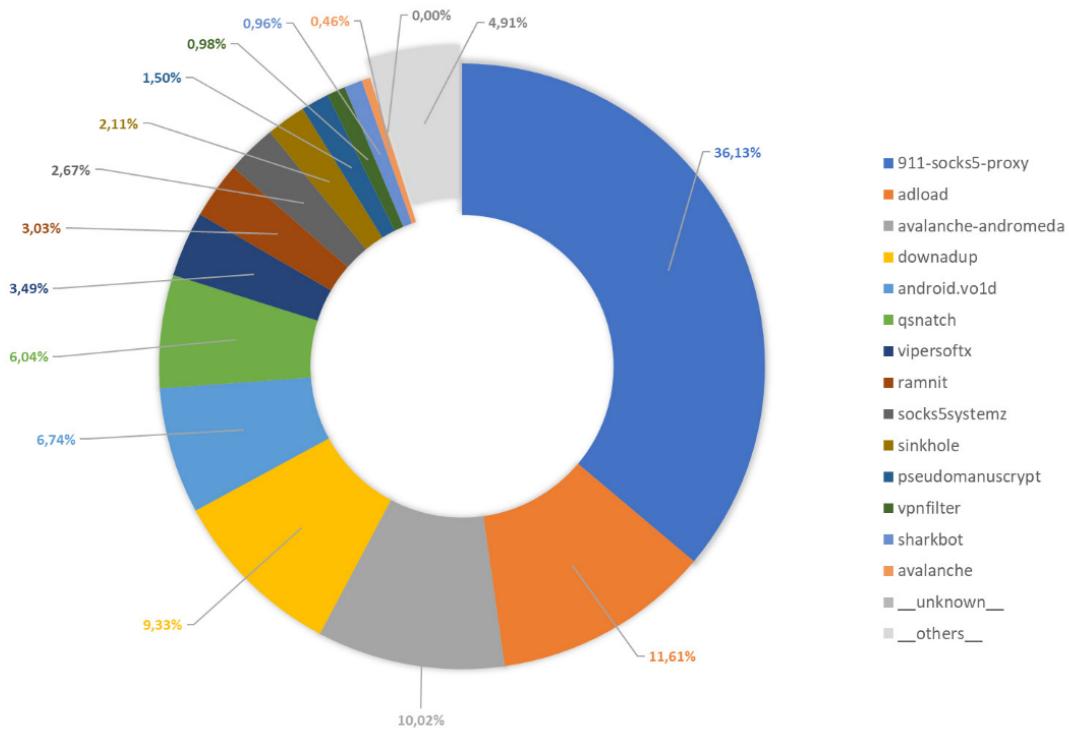


Figura 2.2: Analisi delle infezioni rilevate (dati Fastweb relativi all'anno 2024) [1].

Un altro elemento chiave della difesa a livello locale sono i firewall basati sull'host. A differenza dei firewall di rete, che agiscono a livello di infrastruttura, i firewall locali filtrano i dati in entrata e in uscita del traffico del dispositivo – ad esempio Windows Defender Firewall per Windows o iptables per Linux. Questi consentono di definire regole granulari per ciascuna applicazione o porta, rafforzando il perimetro di sicurezza del sistema.

Le soluzioni, sempre più diffuse per proteggere la comunicazione in rete, sono le VPN (Virtual Private Network): stabiliscono un tunnel cifrato tra il dispositivo dell'utente e un server remoto, nascondendo l'IP e inoltrando tutto il traffico tramite tale canale protetto. Sebbene la maggior parte di tutte le comunicazioni moderne sia ormai protetta con protocolli cifrati – ad esempio il protocollo HTTPS –, utilizzare una VPN aggiunge un ulteriore livello di protezione: l'intero traffico di rete, inclusi i protocolli non cifrati e i metadati DNS, è protetto da intercettazioni a livello di rete, come quelle effettuate da ISP (Internet Service Provider), hotspot Wi-Fi compromessi o altri attori della rete. Questo è particolarmente utile quando si sta lavorando in ambienti pubblici, come Wi-Fi pubbliche, o altre situazioni a rischio come il lavoro remoto. Il National Cyber Threat Assessment 2025–2026 [6] elenca l'ampia adozione delle VPN come pratica di mitigazione del rischio informatico, in particolare per gli utenti mobili.

2.3 Sistemi di difesa a livello di rete

Il panorama dei sistemi di difesa a livello di rete è composto da varie tecnologie e approcci, progettati per proteggere l'intera infrastruttura da minacce esterne ed interne; questi sistemi operano a un livello più ampio rispetto alle soluzioni locali, mirando ad intercettare il traffico malevolo prima che raggiunga i dispositivi finali.

- **Firewall (*Cisco* [25]):** Dispositivi o software che filtrano il traffico di rete in base a regole predefinite, bloccando o consentendo il transito di pacchetti specifici in base ai criteri di sicurezza impostati dall'amministratore di rete. Possono lavorare sia a livello IP/Trasporto (L3/4) o a livelli più alti — come a livello Application (L7) dello stack ISO/OSI — per analizzare il tipo di contenuto dei pacchetti (HTTP, DNS, SMTP, ecc.) (Figura 2.3).

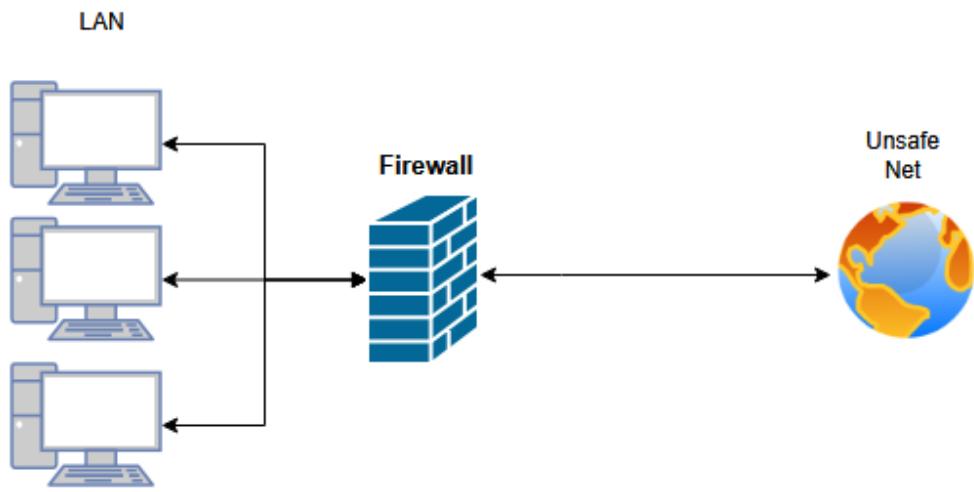


Figura 2.3: Schema di installazione per un dispositivo Firewall.

- **IDS (Intrusion Detection System - *Fortinet* [22]):** Sistemi passivi che monitorano il traffico di rete per identificare attività sospette o non autorizzate, generando allarmi quando vengono rilevati potenziali attacchi. Questi sistemi possono confrontare il traffico con un database di minacce conosciute (firme), oppure basandosi su anomalie, che identificano comportamenti insoliti rispetto a un profilo di traffico normale.
- **IPS (Intrusion Prevention Systems - *Fortinet* [24]):** Simili agli IDS, ma con la capacità di bloccare attivamente il traffico sospetto, impedendo che raggiunga ai dispositivi finali. Spesso vengono integrati con sistemi firewall per fornire una protezione a più livelli (Figura 2.4).

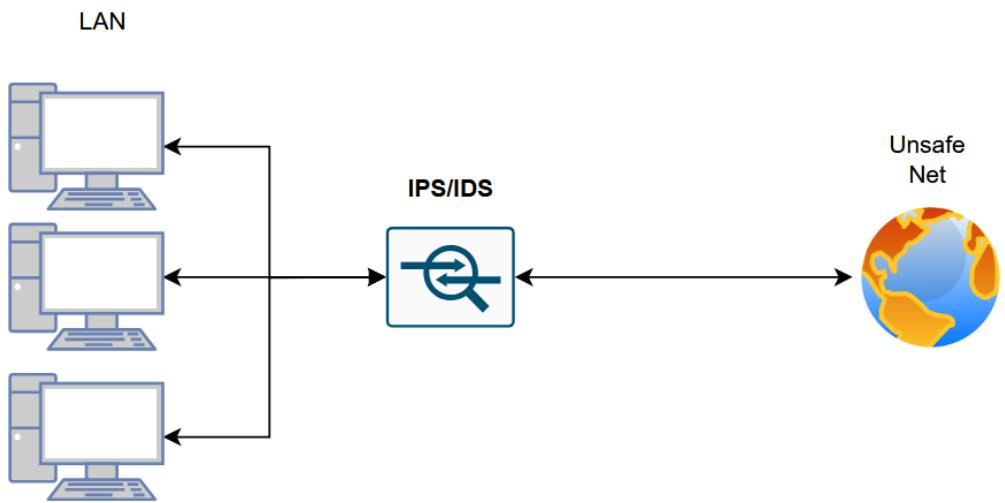


Figura 2.4: Schema di installazione per un dispositivo IPS/IDS.

- **Filtri DNS (*Cloudflare* - [28]):** Sistemi che filtrano le richieste DNS per bloccare l'accesso a domini malevoli o sospetti. Sono spesso implementati per essere raggiungibili direttamente dai client che compongono la rete, garantendo che i dispositivi connessi non riescano a risolvere i nomi di dominio associati a minacce conosciute. Questo approccio non necessita la cattura dei pacchetti in transito, ma si basa sulla configurazione dei client per utilizzare un server DNS specifico, che applica le regole di filtraggio (Figura 2.5).

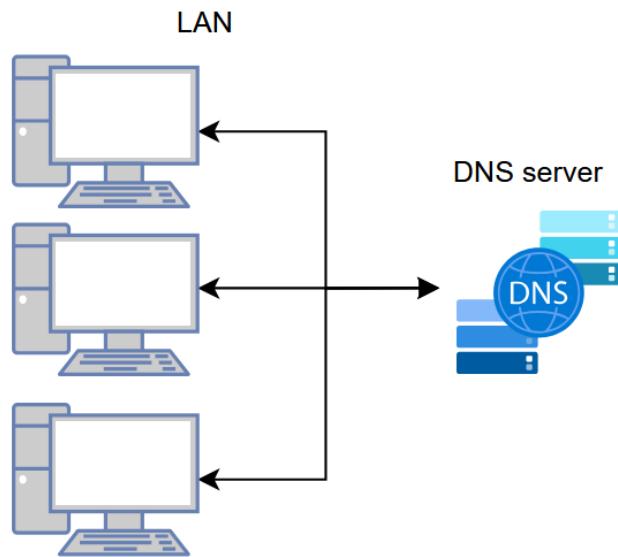


Figura 2.5: Schema di installazione per Server DNS filtrante.

- **SSL Inspection** (*Cloudflare - [29]*): Tecnologie che decriptano il traffico cifrato dal layer TLS (come HTTPS) per ispezionarlo alla ricerca di minacce, consentendo ai sistemi di difesa di analizzare contenuti criptati; si tratta di una pratica molto efficace, ma che implica la gestione di certificati e chiavi private, sollevando preoccupazioni sulla privacy degli utenti. In un contesto Zero Trust è preferibile che questa operazione venga effettuata sui singoli client, piuttosto che sui dispositivi di rete, per garantire una maggiore sicurezza e controllo sui dati sensibili, scongiurando inoltre l'uso di tale implementazione per effettuare attacchi man-in-the-middle (MITM) sull'intera infrastruttura.

L'insieme di questi sistemi fornisce una protezione multilivello, permettendo la raccolta di molte informazioni aggregabili, essenziali per la definizione dello stato della rete e per l'identificazione tempestiva di nuove eventuali minacce. La sicurezza nelle reti grandi e distribuite, come quelle di aziende multinazionali e corporate, è interamente basata sulla raccolta e centralizzazione delle informazioni di threat intelligence, in modo che possano essere analizzate in cerca di pattern malevoli e anomalie, che potrebbero indicare un attacco in corso o imminente. Se si volesse quindi definire uno schema molto semplificato, si potrebbe considerare la seguente rappresentazione come un esempio di partenza (Figura 2.6):

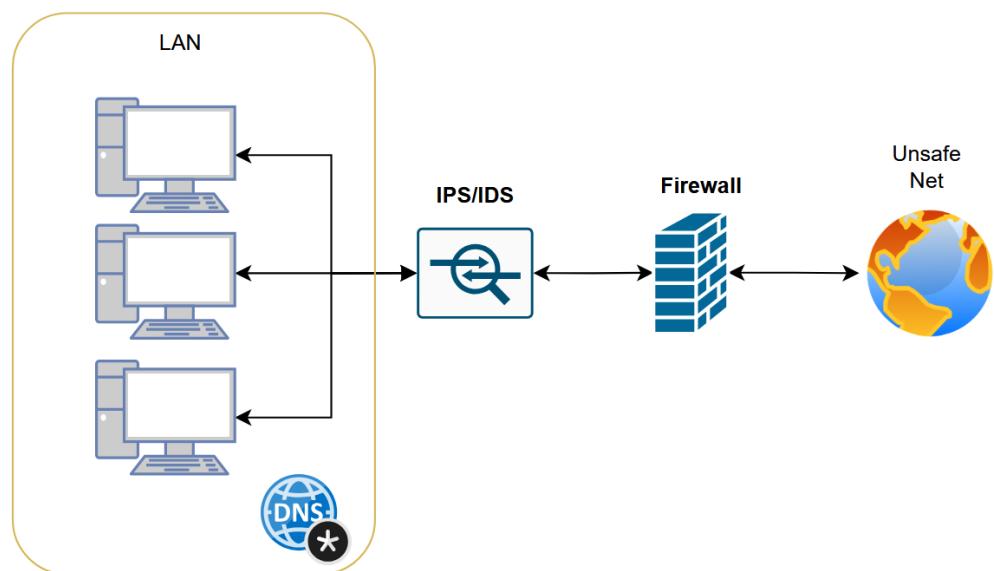


Figura 2.6: Possibile configurazione di un sistema di difesa a livello di rete.

2.4 Verso un approccio integrato: sinergie tra difese locali e di rete

I sistemi di difesa a livello locale (installati sui singoli client) e a livello di rete (implementati nell’infrastruttura di rete) non sono mutualmente esclusivi, ma piuttosto complementari: se un email di phishing può essere neutralizzata con efficacia da un filtro DNS, un antivirus installato sul client può comunque fornire una protezione aggiuntiva contro un malware che potrebbe essere stato scaricato inavvertitamente. In un contesto Zero Trust, dove si presume che ogni dispositivo e utente possa essere compromesso, è fondamentale adottare un approccio misto, che combini le difese locali e di rete per garantire una protezione completa.

2.4.1 Vantaggi di un approccio integrato

Anche se un’infrastruttura di rete ben difesa possa sembrare sufficiente per proteggere i client, in realtà, le minacce possono provenire da molteplici fonti, inclusi supporti esterni come USB o dispositivi mobili; per un corretto approccio Zero Trust, è necessario quindi considerare ogni client come un potenziale punto di ingresso per le minacce, e quindi dotarlo di difese adeguate. Se i client hanno la necessità di installare software, per quanto sicuro possa sembrare, è sempre possibile che un aggiornamento o una configurazione errata possano introdurre vulnerabilità, oppure che uno sviluppatore malintenzionato possa inserire codice malevolo identificabile solo da un’attenta analisi comportamentale da parte di un antivirus.

2.4.2 Il caso SSL

Un esempio di come le difese locali e di rete possano lavorare insieme è la gestione del traffico cifrato: come detto in precedenza, l’uso di tecnologie di SSL Inspection può essere implementato a livello di rete, ma è preferibile lasciare al client la gestione del traffico cifrato; per garantire quindi la sicurezza del dispositivo, si potrebbe decidere di analizzare i file non appena vengano scaricati, in modo da verificarne la sicurezza senza dover rompere la cifratura del traffico in rete.

Capitolo 3

Implementazione delle difese locali

3.1 Introduzione e obiettivi del componente client

In risposta al crescente numero di minacce informatiche legate alla navigazione online, il presente progetto propone lo sviluppo di un'applicazione client che integri funzionalità di connessione VPN basate sul protocollo WireGuard, affiancate da misure di sicurezza locali come la scansione antivirus automatica e la quarantena dei file potenzialmente dannosi.

Lo scopo principale del componente client è garantire che l'utente abbia un'esperienza di navigazione ben controllata e sicura, fornendo al contempo una gestione semplificata della connessione VPN.

Il client fornisce all'utente finale non solo la possibilità di configurare e monitorare un tunnel VPN attraverso un'interfaccia grafica intuitiva, ma anche quella di controllare in tempo reale i file scaricati attraverso tale connessione, eseguire scansioni antivirus tramite *ClamAV* e automatizzare il processo di isolamento dei file ritenuti sospetti (Figura 3.1). A supporto di questa funzionalità, vengono utilizzate le Access Control List (ACL) di Windows per limitare l'accesso ai file in quarantena, rafforzando ulteriormente la sicurezza del sistema.

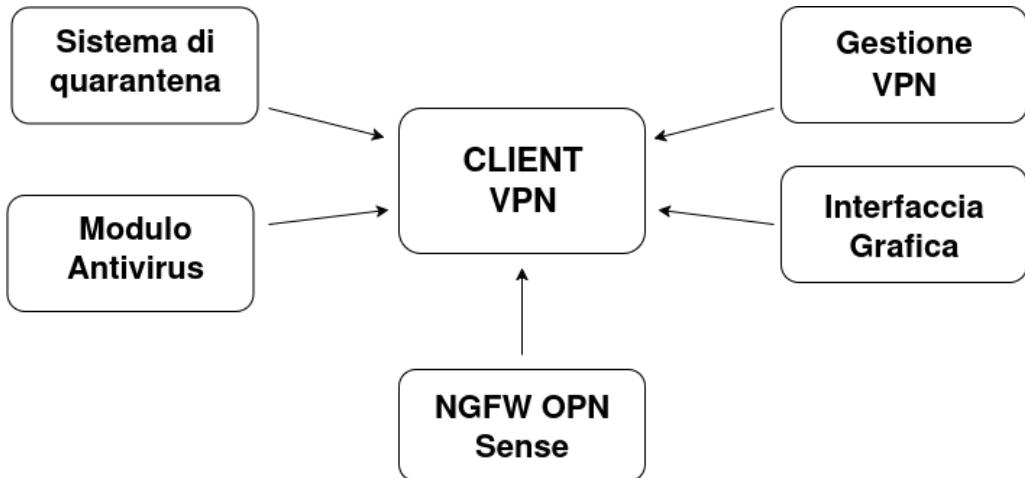


Figura 3.1: Rappresentazione schematica del componente client.

In particolare, gli obiettivi di questa sezione sono:

- Descrivere l'architettura software del componente client e le tecnologie impiegate.
- Illustrare l'integrazione del modulo antivirus, evidenziando il processo di scansione e aggiornamento delle firme malware.
- Presentare il sistema di quarantena per i file potenzialmente dannosi, incluse le modalità di isolamento, gestione e ripristino.
- Spiegare l'implementazione e gestione delle connessioni VPN.
- Discutere lo sviluppo dell'interfaccia utente, con un focus sulla progettazione e sull'implementazione.

Attraverso uno sviluppo modulare e mettendo in primo piano la sicurezza, il progetto mira a colmare le lacune presenti nei software VPN tradizionali proponendo una soluzione all-in-one completamente open-source che integri privacy, sicurezza e usabilità.

Il codice sorgente completo relativo allo sviluppo del componente client è disponibile pubblicamente al seguente repository GitHub:

<https://github.com/LorenzoGallizzioli/WireShield>.

3.2 Architettura software del componente client

L'architettura del software client è realizzata con un approccio a strati, il che permette una separazione di responsabilità tra i componenti chiave dell'architettura secondo lo stile MVC (Model-View-Controller). Questo approccio fa sì che la logica applicativa (Model), l'interfaccia utente (View), e l'orchestratore (Controller) siano indipendenti tra loro (Figura 3.2).

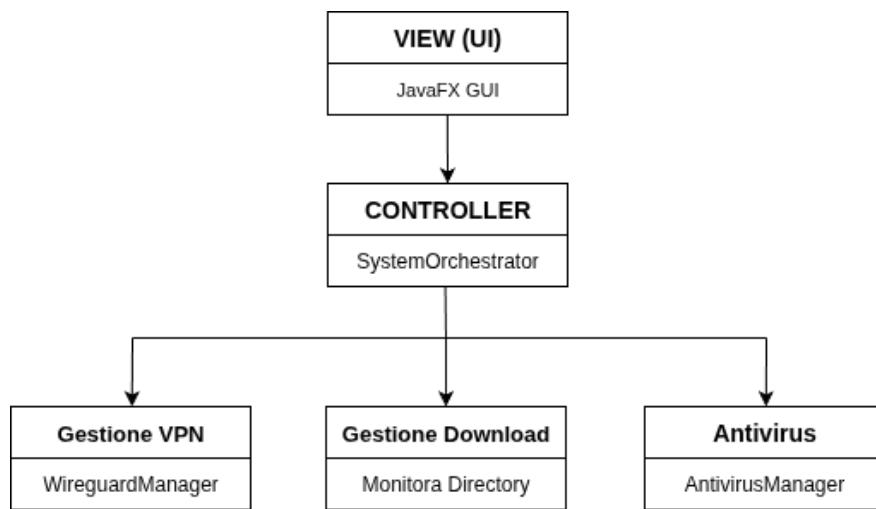


Figura 3.2: Rappresentazione schematica dell'architettura del componente client.

- **Model (gestione della logica e dei dati):**
 - **Modulo di gestione VPN:** Basato sul protocollo *WireGuard*, questo modulo si occupa della configurazione, attivazione e monitoraggio della connessione VPN. Rispetto all'interfaccia standard di *WireGuard* su Windows, che si limita alla gestione di file di configurazione (.conf), il nostro client offre un'interfaccia più moderna che semplifica l'uso e integra il monitoraggio in tempo reale. La gestione della connessione è incapsulata nella classe *WireguardManager*, contenuta nel package *com.wireshield.wireguard*, responsabile della configurazione e gestione sicura dei tunnel VPN.
 - **Modulo di gestione dei download locali:** Monitora in tempo reale le directory di destinazione, rilevando i file scaricati dall'utente durante la

sessione VPN e avviando automaticamente la scansione. Il codice risiede nel package *com.wireshield.localfileutils*.

- **Modulo antivirus locale:** Integra *ClamAV* per la scansione automatica dei file scaricati durante una sessione VPN. *ClamAV* aggiorna regolarmente il suo sistema di firme malware, assicurandosi che le nuove minacce siano sempre rilevate in modo efficace. Il modulo è coordinato dalla classe *AntivirusManager*, contenuta nel package *com.wireshield.av*.
- **Sistema di quarantena:** Quando un file è contrassegnato come dannoso, viene conservato in una zona sicura, implementando le ACL di Windows e gli attributi nascosti su di esso. Il sistema di quarantena permette la gestione di file sospetti con la possibilità di ripristinarli oppure eliminarli in modo permanente. La gestione della quarantena è gestita dalla classe *AntivirusManager*, contenuta nel package *com.wireshield.av*.

- **View (interfaccia utente)**

- **Interfaccia utente (UI):** L’interfaccia utente è sviluppata con JavaFX e si trova nel package *com.wireshield.ui*. Essa fornisce all’utente un controllo completo e intuitivo sia della connessione VPN che delle attività di sicurezza, visualizzando lo stato delle scansioni, le notifiche di quarantena con relative azioni da intraprendere e le impostazioni generali del client.

- **Controller (orchestrazione e coordinazione)**

- **Controller e orchestrazione:** Il coordinamento tra i vari moduli è affidato alla classe *SystemOrchestrator*, contenuta nel package *com.wireshield.localfileutils*. Essa gestisce gli eventi generati dall’utente e invoca i servizi appropriati, fungendo da ponte tra vista e logica applicativa.

Ogni modulo è progettato per essere logicamente indipendente, ma è integrato in modo centralizzato attraverso un orchestratore di sistema, che gestisce il ciclo di vita e le interazioni tra VPN, antivirus e monitoraggio dei download. Questa architettura aiuta a garantire la manutenibilità e facilita eventuali estensioni future.

3.2.1 Approccio asincrono e gestione dei thread

Per garantire che l’interfaccia utente rimanga sempre fluida e reattiva, l’intera applicazione è costruita su un modello di esecuzione asincrono. Tutte le operazioni potenzialmente lunghe o bloccanti sono delegate a thread di controllo dedicati, disaccoppiando così le operazioni di backend dal thread principale della UI di JavaFX. Ogni componente a lunga esecuzione implementa la propria logica di threading:

- Il **WireguardManager** gestisce diversi thread, tra cui `setUpWFPRulesThread` per l’applicazione delle regole del firewall, `startUpdateConnectionStatsThread` per il monitoraggio periodico delle statistiche e `updateWireguardLogsThread` per il recupero continuo dei log, garantendo che l’avvio e il controllo della VPN non blocchino l’interfaccia.
- L’**AntivirusManager** utilizza uno `scanThread` per elaborare in background la coda di file da analizzare (`scanBuffer`), permettendo all’applicazione di continuare a funzionare senza attese durante le scansioni.
- Il **DownloadManager** impiega un `monitorThread` per ascoltare in modo non bloccante gli eventi del file system tramite il `WatchService`.
- La classe **UserInterface** stessa lancia i thread per l’aggiornamento dinamico delle viste, come `logUpdateThread` e `connectionInfoUpdateThread`, per recuperare e visualizzare i dati senza ”congelare” l’applicazione.

Tutte le interazioni che da questi thread di background devono modificare l’interfaccia grafica sono gestite in modo sicuro attraverso chiamate al metodo `Platform.runLater()`, rispettando il modello di threading di JavaFX e garantendo la stabilità del sistema.

3.3 Integrazione del modulo di scansione antivirus locale

L'integrazione di un modulo di scansione antivirus locale consente di aumentare il livello di sicurezza dell'applicazione, offrendo una protezione aggiuntiva che si affianca sia alla privacy garantita dalla connessione VPN, sia alle funzionalità proattive di difesa offerte dal Next-Generation Firewall (NGFW). Questo strato difensivo proattivo ha come obiettivo di proteggere l'utente dalle minacce che possono essere scaricate durante la navigazione in internet e che possono inoltre recare danno al sistema informatico, come **virus**, **trojan**, **worm** e altri tipi di malware. Il modulo in questione è stato progettato e implementato per colpire direttamente il file o i file scaricati, segnalando all'utente le eventuali minacce e permettendo di prendere le opportune decisioni.

3.3.1 Scelta della tecnologia: ClamAV

La scelta di un antivirus richiede di valutarne l'efficacia, l'impatto sulle performance del sistema, i costi e in particolar modo l'allineamento con i principi di privacy e sicurezza del progetto. A seguito di un'analisi comparativa tra soluzioni commerciali (es. Bitdefender) e servizi API-based (es. VirusTotal), la scelta è ricaduta su ClamAV, adottato come principale motore antivirus open-source multipiattaforma per via di diversi fattori strategici:

- **Privacy e controllo locale:** A differenza dei servizi cloud che richiedono l'upload dei file su server di terze parti, ClamAV esegue tutte le scansioni localmente. Questo approccio elimina la necessità di trasmettere dati sensibili a entità esterne, riducendo i potenziali rischi per la privacy dell'utente e garantendo un maggiore controllo sulle proprie informazioni, prevenendo così eventuali data leak.
- **Indipendenza e affidabilità:** Il funzionamento offline di ClamAV garantisce che la protezione sia sempre attiva, indipendentemente dalla stabilità della connessione internet o dalla disponibilità di servizi API esterni. Questa

autonomia lo rende un componente affidabile e sempre disponibile all'interno dell'applicazione.

- **Flessibilità e open source:** ClamAV consente la personalizzazione di diversi parametri attraverso file di configurazione dedicati, come *clamd.conf*, permettendo di adattare il comportamento del motore antivirus alle esigenze dell'applicazione. La natura open-source del progetto garantisce trasparenza e offre la possibilità di analizzarne in dettaglio il funzionamento, rendendolo particolarmente adatto a contesti in cui la comprensione del software è parte integrante della strategia di sicurezza.
- **Costo e accessibilità:** Un fattore determinante per la diffusione di un software è il costo, che nel caso di *ClamAV* non sussiste; questo permette di integrare una funzionalità di sicurezza essenziale senza imporre costi all'utente finale o al progetto stesso.

Sebbene soluzioni commerciali possano offrire tassi di rilevamento superiori contro minacce *zero-day*, per il contesto di questo client — un software sicuro per la navigazione quotidiana e un *proof of concept* accademico — l'efficacia di ClamAV contro le minacce più comuni — come virus, trojan, worm — è ritenuta più che adeguata.

3.3.2 Architettura e implementazione del modulo

Il modulo antivirus è orchestrato dalla classe *AntivirusManager*, che agisce secondo il pattern Singleton, per centralizzare tutte le operazioni di scansione. Essa opera in modo asincrono, utilizzando un buffer interno per accodare i file da analizzare senza bloccare il thread principale dell'applicazione.

Il processo di scansione è quindi gestito da un thread dedicato e segue un flusso operativo ben definito:

1. **Rilevamento e accodamento:** Il *DownloadManager* rileva un nuovo file scaricato e lo inserisce in coda nel buffer condiviso con l'*AntivirusManager*.
2. **Elaborazione asincrona:** Il thread di scansione preleva i file dalla coda in modo sequenziale.

3. **Invocazione della scansione:** Per ogni file, viene invocata la procedura di analisi incapsulata nella classe *ClamAV*. Questa classe gestisce l’interazione con l’engine di ClamAV, eseguendo l’utility a riga di comando *clamdscan.exe* tramite un processo di sistema. L’uso di *clamdscan* sfrutta il demone *clamd* per scansioni più performanti rispetto a *clamscan*.
4. **Analisi dell’output e creazione del report:** L’output del processo viene analizzato per rilevare le parole chiave ”FOUND” o ”SUSPICIOUS”. Il risultato viene incapsulato in un oggetto *ScanReport*, che contiene i dettagli della minaccia e il livello di gravità (*warningClass*).
5. **Azione conseguente:** Se il report indica una minaccia (*DANGEROUS* o *SUSPICIOUS*), *AntivirusManager* avvia immediatamente il protocollo di quarantena. In caso contrario, il file è considerato sicuro.

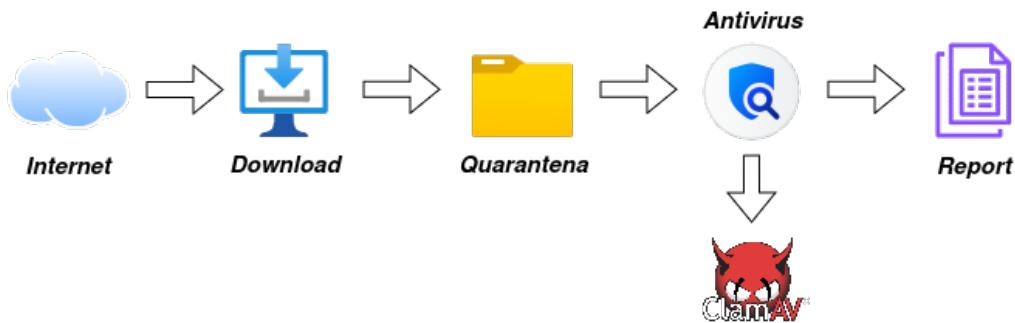


Figura 3.3: Flusso di lavoro del modulo antivirus.

Questa architettura disaccoppiata e asincrona garantisce che l’interfaccia utente rimanga reattiva anche durante la scansione di file di grandi dimensioni, delegando l’interazione con il motore antivirus a componenti specializzati (Figura 3.3).

3.3.3 Gestione degli aggiornamenti e delle firme malware

L’efficacia di un antivirus basato su firme dipende criticamente dall’aggiornamento del suo database. Per garantire che ClamAV sia sempre in grado di riconoscere le minacce più recenti, l’applicazione integra un meccanismo di aggiornamento automatico. Questo compito è delegato a **FreshClam**, l’utility ufficiale di ClamAV.

La classe *ClamAV* incapsula una funzionalità per avviare il processo *freshclam.exe* in un thread separato. Questo permette all'applicazione di aggiornare le firme dei malware in background, senza impattare l'esperienza utente e implementando l'esecuzione di azioni successive al completamento dell'operazione, come notificare l'utente dell'avvenuto aggiornamento.

3.3.4 Implementazione della quarantena e isolamento dei file

La gestione dei file sospetti è una componente critica della strategia di difesa. Anziché eliminare immediatamente un file, che potrebbe essere un falso positivo, il software lo isola in un'area di quarantena sicura.

Il processo è gestito dalla classe *AntivirusManager* e si articola in due fasi principali:

- **Blocco dell'esecuzione:** Come prima misura, il file viene reso non eseguibile tramite una ridenominazione, effettuata da una procedura dedicata interna al *FileManager*, nella quale aggiunge un'estensione *.blocked* al nome del file. Questa tecnica semplice ma efficace impedisce l'apertura accidentale del file da parte dell'utente o di altri processi.
- **Spostamento in quarantena:** Successivamente, il file bloccato viene spostato in una directory nascosta *.QUARANTINE*, creata all'interno della cartella monitorata. Una procedura dedicata dell'*AntivirusManager* gestisce questa operazione e, per una maggiore sicurezza su sistemi Windows, imposta delle *Access Control List (ACL)* sulla directory. Le ACL vengono configurate per garantire all'utente corrente i permessi di lettura, ma negano la scrittura e l'esecuzione, limitando ulteriormente la possibilità che il file possa causare danni.

Per ogni file messo in quarantena, viene creato un file di metadati *.meta* che memorizza informazioni cruciali come il percorso originale, la data e i dettagli della minaccia, facilitando le successive operazioni di ripristino o eliminazione definitiva gestite dall'interfaccia utente.

Approccio "safety-first": quarantena preventiva

Una scelta architetturale cruciale riguarda l'ordine delle operazioni, infatti, non appena un file stabile viene rilevato, non viene immediatamente scansionato, ma, al contrario, il flusso operativo diviene il seguente:

1. Il *DownloadManager* rileva il nuovo file.
2. Invoca immediatamente l'*AntivirusManager* per spostare il file in quarantena.
3. Solo dopo che il file è stato isolato con successo, viene aggiunto alla coda di scansione dell'antivirus.

Questo approccio "safety-first" assicura che un file potenzialmente dannoso venga confinato in un'area sicura prima ancora che l'analisi vera e propria abbia inizio, massimizzando la protezione del sistema.

3.4 Implementazione del modulo di monitoraggio dei download

La strategia difensiva dell'applicazione inizia con il monitoraggio automatico dei download. Il suo scopo è agire come una sentinella, rilevando in tempo reale la creazione di nuovi file all'interno di una directory specifica per sottoporli immediatamente al processo di scansione antivirus. Questo componente è essenziale per automatizzare la protezione, garantendo che nessun file scaricato sfugga al controllo prima di poter essere eseguito.

3.4.1 Obiettivi e architettura

Il modulo di monitoraggio dei download è stato progettato per essere altamente leggero, efficiente, non intrusivo e indipendente dagli altri servizi dell'applicazione. Il componente centrale è la classe *DownloadManager*, implementata secondo il pattern Singleton, per assicurare che un'unica istanza sia responsabile del monitoraggio, evitando così conflitti e spreco di risorse.

Le responsabilità del *DownloadManager* sono:

- Avviare e interrompere il monitoraggio di una directory configurabile.
- Rilevare la creazione o la modifica di file al suo interno.
- Eseguire controlli preliminari per scartare file incompleti o temporanei.
- Inoltrare i file validi al modulo antivirus per l'analisi.

Per non bloccare il thread principale dell'interfaccia utente, l'intera logica di monitoraggio viene eseguita in un thread separato in background. Questo approccio asincrono garantisce che l'applicazione rimanga sempre reattiva, anche se il file system genera un gran numero di eventi in rapida successione.

3.4.2 Implementazione tecnica: il WatchService di Java NIO

La tecnologia alla base del monitoraggio è **WatchService**, un'API introdotta nel package Java NIO (New I/O). Questa scelta è strategica, in quanto offre un meccanismo di monitoraggio degli eventi del file system basato su notifiche native del sistema operativo, anziché su un approccio di *polling* attivo, che sarebbe invece inefficiente e comporterebbe un consumo di CPU non necessario.

Il processo di implementazione si articola nei seguenti passaggi:

1. Al momento dell'avvio del servizio di monitoraggio, viene creata un'istanza del *WatchService*.
2. La directory da monitorare viene "registrata" presso il servizio, specificando gli eventi di interesse, in particolare *ENTRY_CREATE* (creazione di un nuovo file) ed *ENTRY_MODIFY* (modifica di un file esistente).
3. Il thread dedicato entra in un ciclo di attesa, bloccandosi in modo efficiente finché il servizio non notifica la presenza di uno o più eventi.
4. Una volta ricevuta la notifica, il thread elabora gli eventi, identifica i file coinvolti e avvia le procedure di sicurezza.

3.4.3 Gestione degli eventi e mitigazione dei rischi

La semplice rilevazione di un evento di creazione non è sufficiente per garantire un'analisi corretta. Sono state implementate diverse logiche per gestire scenari comuni e prevenire errori.

Gestione delle scansioni premature

Durante la fase di download, i file non vengono salvati immediatamente nella loro forma completa, ma vengono scritti progressivamente sul disco man mano che i dati vengono ricevuti.

Browser come Google Chrome, ad esempio, utilizzano file temporanei con estensioni come `.crdownload` per rappresentare questi file in corso di scaricamento. Poiché tali file sono ancora incompleti, una scansione antivirus eseguita in questa fase risulterebbe poco affidabile: il contenuto analizzato non corrisponde al file finale, il che potrebbe impedire l'identificazione di minacce reali o, al contrario, generare falsi positivi.

La sola verifica del nome del file, tuttavia, non è sufficiente; al termine del download, infatti, il browser prima rinomina il file al suo nome definitivo e, solo un istante dopo, finalizza la scrittura su disco. Questo crea una potenziale *race condition*, in cui il sistema potrebbe rilevare il file con il nome corretto ma tentare di scansionarlo mentre è ancora incompleto o bloccato dal processo di scrittura.

Per mitigare questo duplice rischio ed eseguire la scansione solo una volta completato interamente il download, *DownloadManager* esegue due controlli prima di procedere:

- **Verifica del nome del file:** Ignora i file con estensioni note come temporanee.
- **Verifica della stabilità del file:** Introduce una breve pausa per assicurarsi che nessun processo stia più scrivendo attivamente sul file, verificandone la stabilità.

Prevenzione di scansioni multiple

Un singolo download può generare più eventi di file system in rapida successione, di conseguenza, per evitare che lo stesso file venga messo in quarantena e scansionato più volte, il *DownloadManager* mantiene un set di file già processati.

Per identificare univocamente ogni risorsa, il sistema non si basa sul solo percorso del file, ma su un identificatore composito generato dalla concatenazione del percorso assoluto del file e del suo timestamp di ultima modifica (`lastModified`). Questo approccio è robusto e previene potenziali conflitti: se un file scansionato venisse rimosso e, in un secondo momento, un nuovo file con lo stesso nome venisse scaricato nella medesima posizione, quest'ultimo avrebbe un timestamp di modifica differente. Di conseguenza, verrebbe generato un nuovo identificatore univoco, garantendo che il file venga correttamente riconosciuto come una nuova risorsa da sottoporre ad analisi. Ogni nuovo file rilevato viene aggiunto a questa collezione solo se il suo identificatore non è già presente, garantendo che ogni download venga gestito una sola volta.

3.5 Implementazione e gestione della connessione VPN

Il nucleo operativo dell'applicazione è costituito dal modulo di gestione della connessione VPN, sviluppato per offrire un'interfaccia unica e intuitiva che semplifica l'interazione con il protocollo WireGuard. Questo modulo non si limita a stabilire e interrompere il tunnel, ma gestisce l'intero ciclo di vita della connessione, dal parsing delle configurazioni al monitoraggio in tempo reale delle statistiche di traffico e all'integrazione con il firewall di sistema.

3.5.1 Architettura del componente VPN

Per garantire una chiara separazione delle responsabilità e favorire la manutenibilità del codice, i componenti chiave sono stati organizzati nel modo seguente:

- **WireguardManager:** Questa classe funge da orchestratore centrale del modulo, implementando il design pattern Façade per fornire un'unica interfaccia semplificata che nasconde la complessità dei componenti sottostanti. In questo modo, gli altri moduli dell'applicazione, in particolare l'interfaccia utente, comunicano esclusivamente con il WireguardManager per eseguire operazioni di alto livello, senza doversi preoccupare dei dettagli implementativi della gestione della connessione o dei peer. Implementato come Singleton per garantire un'unica istanza di controllo, il suo compito è quello di coordinare l'intero ciclo di vita della connessione: dall'avvio e arresto del tunnel, alla gestione dei log e all'aggiornamento delle statistiche, delegando ogni operazione specifica ai componenti più specializzati.
- **Connection:** Rappresenta lo stato attuale della connessione di rete. Anche esso implementato con il pattern Singleton, è responsabile della comunicazione diretta con gli eseguibili di WireGuard ('wireguard.exe' e 'wg.exe') tramite processi di sistema. Incapsula la logica per avviare e terminare il servizio del tunnel, interrogare lo stato dell'interfaccia e recuperare dati grezzi come il traffico trasferito e il tempo dell'ultimo handshake.

- **PeerManager e Peer:** Queste due classi lavorano in sinergia per gestire le configurazioni dei profili VPN. Invece di manipolare direttamente il testo grezzo dei file `.conf`, il sistema utilizza la classe `Peer` come un modello strutturato, ovvero un oggetto che rappresenta in modo ordinato tutti i parametri di una singola connessione, come le chiavi crittografiche, l'endpoint del server e gli indirizzi IP consentiti. Il `PeerManager`, a sua volta, agisce da gestore di questi oggetti: si occupa di leggere i file di configurazione, di "tradurli" creando un oggetto `Peer` per ciascuno di essi, e di mantenerli in una collezione interna. In questo modo, il resto dell'applicazione può interagire con i profili in modo semplice e sicuro, richiedendo al `PeerManager` di creare, recuperare o eliminare un peer senza dover mai accedere direttamente ai file su disco.

3.5.2 Gestione del ciclo di vita della connessione

L'avvio e l'arresto del tunnel VPN sono operazioni asincrone, gestite in thread separati per non bloccare l'interfaccia utente e garantire una reattività ottimale.

Avvio della connessione

Quando l'utente richiede di stabilire una connessione, il `WireguardManager` invoca il processo di avvio, il quale, a sua volta, istruisce la classe `Connection` a eseguire il comando `wireguard.exe /installtunnelservice nomefile.conf`. Questo comando nativo di WireGuard crea un servizio di Windows per il tunnel specificato, stabilendo di fatto la connessione VPN. Il sistema attende quindi in modo attivo che l'interfaccia di rete risulti visibile al sistema operativo prima di aggiornare il suo stato interno a "Connesso", notificando il successo dell'operazione (Figura 3.4).

Arresto della connessione

In modo speculare, la disconnessione viene gestita eseguendo il comando `wireguard.exe /uninstalltunnelservice nomeinterfaccia`, che arresta e rimuove il servizio di Windows associato al tunnel, chiudendo la connessione. Anche in questo caso, il

sistema attende la conferma della disattivazione dell’interfaccia prima di considerare l’operazione completata e aggiornare lo stato (Figura 3.5).

Monitoraggio e raccolta delle statistiche

Una volta stabilita la connessione, un thread dedicato, avviato dal *WireguardManager*, si occupa di aggiornare periodicamente le statistiche della connessione. Questo thread interroga a intervalli regolari la classe *Connection*, che a sua volta esegue il comando `wg.exe show nomeinterfaccia` con vari parametri per ottenere:

- **Traffico trasferito:** Utilizzando il parametro *transfer*, si ottengono i byte inviati e ricevuti.
- **Ultimo handshake:** Con il parametro *latest-handshakes*, si verifica la data dell’ultimo handshake, un indicatore fondamentale della salute della connessione.

Questi dati vengono poi resi disponibili nell’interfaccia utente per fornire un feedback in tempo reale sull’attività della VPN. Parallelamente, un altro thread si occupa di recuperare i log diagnostici di WireGuard, eseguendo il comando `/dumplog` e rendendoli disponibili per la visualizzazione.

3.5.3 Gestione delle configurazioni dei peer

Per offrire flessibilità, l’applicazione non è legata a una configurazione fissa, ma può caricare dinamicamente i file *.conf* di WireGuard; infatti, la classe *PeerManager* contiene una procedura di parsing che legge il contenuto di un file di configurazione e, tramite l’uso di espressioni regolari, ne estrae in modo robusto le sezioni (*[Interface]*, *[Peer]*) e le relative coppie chiave-valore. I dati estratti vengono poi utilizzati per istanziare un oggetto *Peer*, che viene aggiunto alla lista dei profili disponibili per l’utente. Questo approccio garantisce la corretta interpretazione dei file anche in presenza di formattazioni diverse.

3.5.4 Integrazione avanzata con il firewall di Windows (WFP) per il kill-switch

Una delle funzionalità di sicurezza più rilevanti implementate è un *kill-switch* avanzato e non disattivabile. Questo componente rappresenta un pilastro fondamentale e obbligatorio della strategia di sicurezza, la cui funzione è quella di forzare ogni singolo pacchetto di rete a transitare attraverso il tunnel VPN, prevenendo i cosiddetti "VPN leaks", ovvero le fughe di dati che possono vanificare completamente la protezione offerta. Senza un meccanismo di blocco robusto, il sistema operativo Windows, nella sua normale ottimizzazione delle connessioni, potrebbe decidere di instradare determinate richieste (in particolare le query DNS) attraverso la connessione di rete più veloce disponibile, bypassando di fatto il tunnel VPN. Per l'applicazione, l'integrazione del kill-switch assume quindi una valenza ulteriore: oltre a garantire la protezione completa del traffico, permette di mantenere l'accesso a specifiche risorse della rete locale (LAN), altrimenti non raggiungibili senza una configurazione manuale delle rotte. La realizzazione di questa funzionalità ha richiesto un'analisi approfondita del funzionamento di WireGuard su Windows a livello di sistema.

Analisi preliminare e limiti degli strumenti standard

Un'indagine preliminare ha dimostrato che le regole di firewall applicate da WireGuard non sono né permanenti né possono essere visualizzate attraverso utility di sistema come `netsh` o i comandi di gestione del firewall in PowerShell. Ciò è dovuto al fatto che le regole in questione sono effimere, applicate in fase di creazione del tunnel e disattivate appena esso viene interrotto, obbligandoci all'utilizzo di software avanzati come *WindowsFirewallPlatformExplorer* per ispezionare le regole applicate.

Studio del codice sorgente di WireGuard-Windows

L'analisi diretta del codice sorgente del client WireGuard per Windows si è rivelata fondamentale per comprendere nel dettaglio le modalità con cui il software interagisce con il firewall, rendendo esplicativi i meccanismi interni di gestione delle regole di rete:

- Le regole vengono create programmaticamente utilizzando le API della libreria Win32 di Microsoft, interagendo direttamente con i *layer* della Windows Filtering Platform.
- Ciascun layer del firewall è identificato da un UUID (Universally Unique Identifier) specifico, il che consente a WireGuard di operare in modo estremamente granulare sullo stack di rete.
- Le regole sono strettamente associate all'interfaccia di rete virtuale creata da WireGuard, la quale viene identificata a livello di sistema da un LUID (Locally Unique Identifier).

Progettazione e implementazione della soluzione

Per far coesistere le funzionalità di kill-switch con la possibilità di garantire la comunicazione locale, è necessario applicare nella Windows Filtering Platform regole specifiche con priorità superiore a quelle di WireGuard. A tal fine, l'approccio individuato prevede l'impostazione di tali regole con la massima priorità, anticipando l'applicazione di quelle definite da WireGuard, che a sua volta tende a impostarle con priorità massima. Questo trucco permette di aggirare il problema, difficilmente risolvibile anche tramite le librerie dinamiche ufficiali (DLL, Dynamic Link Libraries), garantendo che il traffico verso le destinazioni impostate dalle regole non passi per il tunnel VPN. Per attuare questa strategia è stata implementata una chiara separazione di responsabilità: la classe *WFPManger* agisce come un gestore di configurazione, il cui unico compito è preparare il comando per l'eseguibile esterno `main.exe`. Essa legge i CIDR (Classless Inter-Domain Routing) permessi da un file di supporto associato al profilo e li assembla in una stringa di comando completa. Il *WFPManger*, quindi, non avvia alcun processo, ma fornisce solo il comando

pronto per essere eseguito. Queste regole sono costruite "ad-hoc" per consentire il traffico in uscita verso i CIDR impostati, come ad esempio quelli delle reti private (192.168.0.0/16, 10.0.0.0/8 e 172.16.0.0/12). L'esecuzione vera e propria è invece orchestrata dal *WireguardManager*, il quale, al momento della richiesta di connessione, segue le operazioni in una sequenza rigorosa:

1. Invoca il *WFPManager* per ottenere la stringa di comando completa e personalizzata per il profilo selezionato.
2. Avvia l'eseguibile "WFP rules generator" come un processo separato in background, passandogli i CIDR permessi come argomenti.
3. Avvia il servizio del tunnel di WireGuard.

Questa sequenza garantisce che, quando WireGuard tenta di applicare il suo blocco totale, le regole di permesso per la LAN siano già state registrate in WFP e abbiano quindi la precedenza, garantendo che il traffico selezionato non venga instradato nel tunnel VPN. La gestione del ciclo di vita di questo processo esterno è altrettanto fondamentale: quando la connessione viene interrotta, il *WireguardManager* si assicura che il processo del "rules generator" venga terminato. La natura non persistente delle regole WFP create in questo modo garantisce che, alla chiusura del processo che le ha create, esse vengano automaticamente rimosse dal sistema, ripristinando lo stato di rete originale e prevenendo configurazioni anomale.

In sintesi, il flusso di accensione e spegnimento della connessione VPN è illustrato nelle Figure 3.4 e 3.5.



Figura 3.4: Schema riassuntivo del flusso di accensione VPN.



Figura 3.5: Schema riassuntivo del flusso di spegnimento VPN.

3.6 Sviluppo dell’interfaccia utente

La UI (User Interface, o interfaccia utente) è il punto di contatto tra l’utente e le funzionalità di sicurezza e networking introdotte dall’applicativo. La sua progettazione è stata guidata da principi di chiarezza, reattività e coerenza, con l’obiettivo di rendere accessibili e comprensibili operazioni tecniche.

3.6.1 Filosofia di progettazione e tecnologie

Lo sviluppo della UI è basato sul framework **JavaFX**. Al fine di isolare la logica di esecuzione applicativa da quella di presentazione, si è deciso di implementare il pattern **FXML**, un approccio dichiarativo che consente di costruire la struttura dell’interfaccia in un file XML, delegando la logica e il comportamento a un controller Java (nel nostro caso *UserInterface.java*). Lo styling visuale è delegato a un foglio di stile **CSS** esterno (*styles.css*), che permette di modificare l’aspetto visivo dell’applicazione, non richiedendo contestualmente delle modifiche al codice sorgente. Inoltre, per migliorare ulteriormente la leggibilità e l’estetica dell’interfaccia, è stata integrata la libreria **Ikonli**, la quale permette di usare un vastissimo set di icone vettoriali, come FontAwesome, rendendo l’esperienza più chiara e gradevole.

3.6.2 Architettura della vista principale

L’applicazione adotta un’architettura a ”Single-Pane”, in cui una singola finestra contiene più viste che vengono mostrate o nascoste in base all’interazione dell’utente. Il ruolo di **Controller** è ricoperto dalla classe *UserInterface.java*, la quale è direttamente collegata agli elementi della vista (pulsanti, etichette, pannelli) e ha due responsabilità principali:

1. **Gestire gli eventi della UI:** Intercetta le azioni dell’utente, come il click su un pulsante o la selezione di un profilo.
2. **Fare da mediatore verso il backend:** Non contiene la logica complessa (es. come avviare una VPN o scansionare un file), ma traduce le azioni dell’utente in comandi da inviare allo strato successivo (*SystemOrchestrator.java*).

Come detto nelle righe precedenti, il controller *UserInterface.java* comunica esclusivamente con il *SystemOrchestrator.java*, che rappresenta il punto di ingresso al **Model** dell'applicazione. In questo modo, il *UserInterface.java* inoltra le richieste (es. "avvia la VPN con questo profilo") all'orchestratore, il quale si occuperà poi di coordinare i componenti specifici del backend (come il *WireguardManager* o l'*AntivirusManager*) per eseguire l'operazione richiesta. Questa netta separazione garantisce che l'interfaccia utente rimanga disaccoppiata dalla logica dell'applicazione, migliorando la modularità e la manutenibilità del sistema.

I pannelli principali (**AnchorPane**), gestiti dal Controller *UserInterface.java*, sono presentati nell'ordine in cui appaiono nell'interfaccia:

- **Home pane:** La dashboard principale, che mostra lo stato della connessione VPN e una lista dei profili (peer) disponibili per la connessione.
- **AV pane:** Presenta i risultati delle scansioni antivirus, lo stato del servizio di protezione e le azioni disponibili per i file analizzati.
- **Logs pane:** Visualizza i log diagnostici in tempo reale provenienti da WireGuard, utili per il troubleshooting e il monitoraggio avanzato.
- **Settings pane:** Permette all'utente di configurare parametri dell'applicazione — come la cartella da monitorare per i download.

3.6.3 Gestione asincrona e aggiornamento dei dati

Per garantire che l'interfaccia utente rimanga sempre fluida e reattiva, tutte le operazioni di recupero dati che potrebbero richiedere tempo sono eseguite in background su thread separati. Il controller principale gestisce diversi cicli di aggiornamento asincroni:

- **Aggiornamento dello stato VPN:** Un thread dedicato controlla lo stato della connessione a intervalli regolari e aggiorna le etichette riguardanti il traffico inviato/ricevuto e il tempo trascorso dall'ultimo handshake. I dati numerici vengono formattati in modo leggibile attraverso una classe di utilità (*TimeUtil.java*).

- **Aggiornamento dei log:** Un altro thread si occupa di leggere periodicamente i log di WireGuard e di aggiornare la **TextArea** dedicata, permettendo all'utente di monitorare l'attività di basso livello senza bloccare l'applicazione (Figura 3.6).

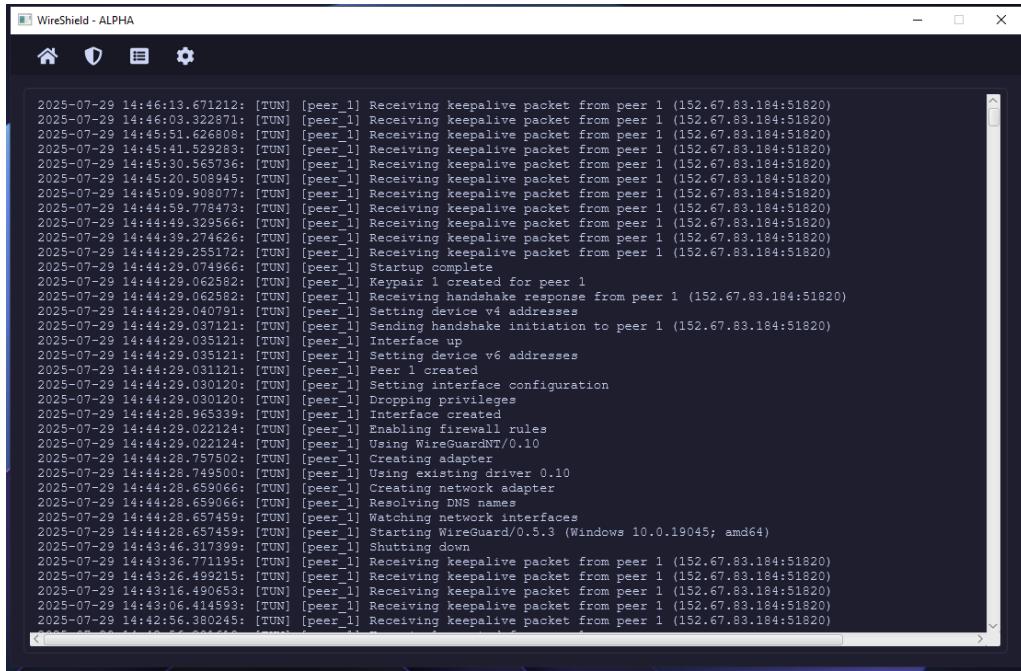


Figura 3.6: Immagine illustrativa dei log.

- **Aggiornamento dei report antivirus:** La vista antivirus viene aggiornata da un thread che recupera la lista dei report di scansione e li visualizza dinamicamente, garantendo che i risultati appaiano non appena disponibili.

Tutte le modifiche ai componenti della UI provenienti da questi thread avvengono tramite chiamate a *Platform.runLater()*, la pratica standard in JavaFX per garantire la sicurezza e la stabilità delle operazioni sull'interfaccia grafica.

3.6.4 Progettazione a componenti riutilizzabili

Per migliorare la manutenibilità e la leggibilità del codice, l'interfaccia è stata suddivisa in componenti logici e riutilizzabili, ciascuno con la propria logica e, in alcuni casi, con il proprio controller.

Schede informative per i file scansionati

Per presentare i risultati delle scansioni antivirus in modo chiaro e organizzato, la vista non utilizza una semplice lista testuale, ma genera dinamicamente una serie di schede informative (o "card"), ciascuna dedicata a un singolo file analizzato. La logica di queste schede è incapsulata nella classe *FileCardComponent.java*, un elemento grafico personalizzato che estende un *VBox* di JavaFX. Ogni card è progettata per fornire un riepilogo immediato dello stato del file, mostrando:

- Un **indicatore di stato** circolare e colorato, che permette di identificare a colpo d'occhio la pericolosità del file (verde per innocuo, giallo per sospetto, rosso per pericoloso).
- Il nome e il percorso del file, per un'identificazione inequivocabile.
- Un **pannello di dettagli espandibile**, che l'utente può aprire per visualizzare informazioni aggiuntive come la data e l'ora esatta della scansione (Figura 3.7).

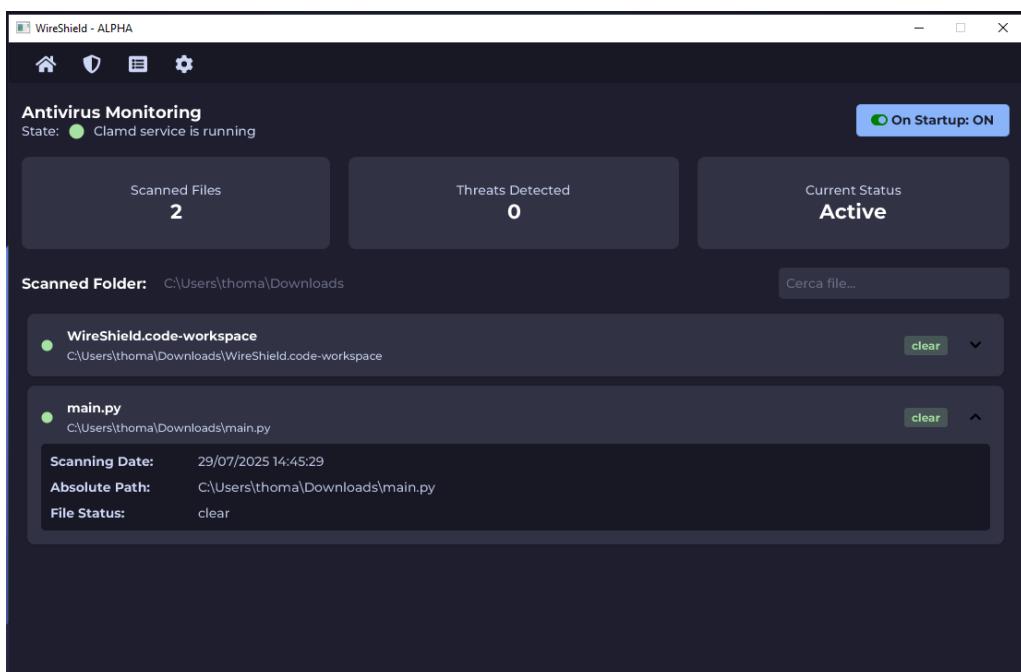


Figura 3.7: Immagine illustrativa dei dettagli di un file scansionato.

- Una serie di **pulsanti di azione** (es. "Ripristina", "Elimina") che appaiono dinamicamente solo quando necessari, ovvero quando un file viene identificato come sospetto o pericoloso e richiede un intervento da parte dell'utente (Figura 3.8).

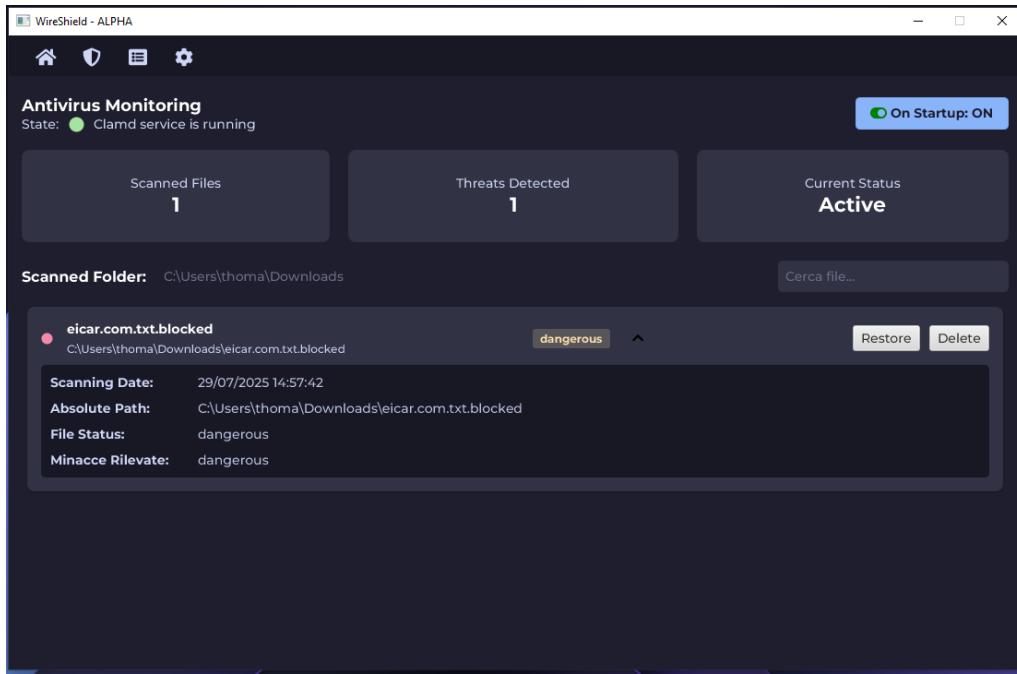


Figura 3.8: Immagine illustrativa delle opzioni disponibili per un file infetto.

Così, ogni volta che il controller principale riceve un nuovo *ScanReport.java* dal backend, non si limita a visualizzare i dati, ma crea direttamente una scheda interattiva, riempiendo la vista in modo modulare e offrendo all'utente un'esperienza sia gradevole che utile.

Pannello di gestione dei peer

Per mantenere l'interfaccia utente organizzata e modulare, la vista principale è stata strutturata in due sezioni verticali distinte: la sezione sinistra mostra la lista di tutti i profili VPN disponibili, mentre la sezione destra, inizialmente vuota, è dedicata a visualizzare i dettagli del profilo selezionato (Figura 3.9). La gestione dei dettagli di un profilo VPN non è implementata direttamente nel controller principale *UserInterface.java*, ma è delegata a una sotto-vista, ovvero un componente UI indipendente,

definito in un file FXML separato (`peerInfo.fxml`). Poiché questa sotto-vista ha elementi interattivi propri (etichette, pulsanti, campi di testo), richiede un proprio controller per gestirne la logica specifica. Questo ruolo è ricoperto dalla classe `PeerInfoController.java`, la quale si occupa di popolare i campi con i dati del profilo e di intercettare le azioni dell’utente all’interno di quel pannello. In questo modo, si crea una netta separazione di responsabilità: il controller principale (`UserInterface.java`) si occupa della gestione generale della finestra, mentre il `PeerInfoController.java` si concentra esclusivamente sulla visualizzazione dei dati del profilo VPN selezionato e sulla gestione delle sue funzionalità avanzate — come la configurazione delle regole CIDR per il kill-switch.

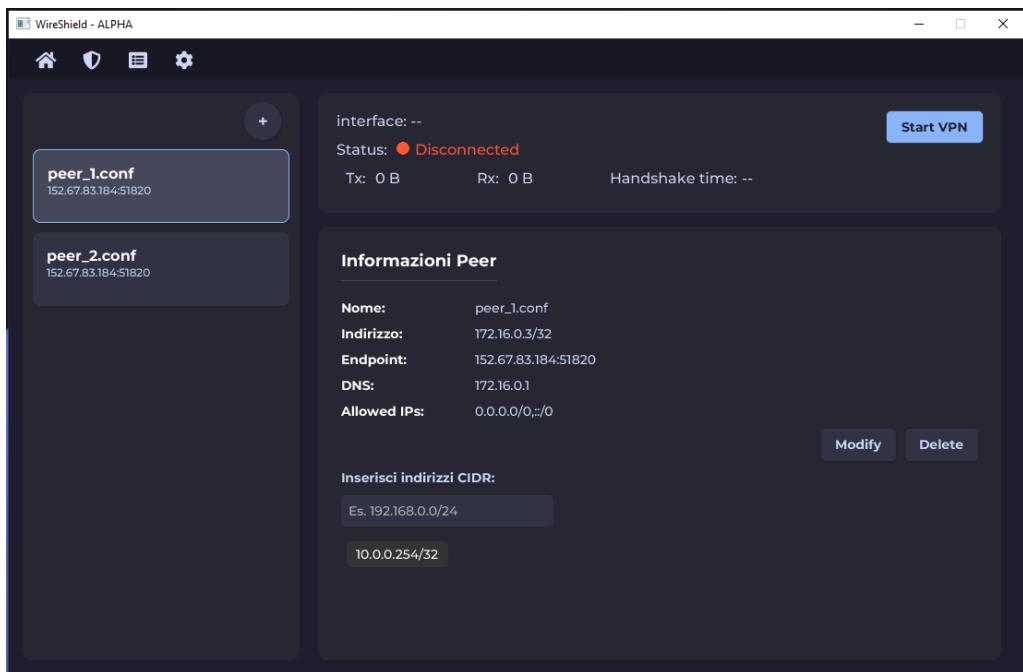


Figura 3.9: Immagine illustrativa della vista del pannello di gestione dei peer.

Questa architettura a ”controller annidati” solleva una sfida: come può il `PeerInfoController.java`, responsabile della sotto-vista, notificare un’azione dell’utente (es. ”elimina questo peer”) alla classe `UserInterface.java`, che gestisce la vista principale ed è l’unica a poter orchestrare l’operazione a livello di applicazione? Per risolvere questo problema in modo pulito e disaccoppiato, è stato implementato il design pattern **listener/callback**.

Il meccanismo funziona così:

1. È stata definita un'interfaccia, *PeerOperationListener.java*, che descrive le azioni possibili (es. *onPeerDeleted*).
2. Il Controller principale implementa questa interfaccia, rendendosi di fatto un "ascoltatore" di questi eventi.
3. Quando il Controller principale crea l'istanza del *PeerInfoController.java*, gli passa un riferimento a se stesso come listener.
4. A questo punto, quando l'utente clicca sul pulsante "Elimina" nel pannello dei dettagli, il *PeerInfoController.java* non tenta di eseguire l'azione direttamente, ma si limita a notificare il suo listener, dicendo: "l'utente vuole eliminare questo peer".

Dopo la ricezione della notifica, il Controller principale chiama il *SystemOrchestrator.java* per procedere con la cancellazione effettiva. In questo modo si preserva la logica dell'applicazione in un punto unico, mentre la UI resta disaccoppiata e riutilizzabile, aumentando la flessibilità e la possibilità di estensione. I controller si scambiano informazioni con semplici invocazioni di metodo, evitando la gestione diretta dei thread, che è invece responsabilità esclusiva dei blocchi di backend (Figura 3.10).

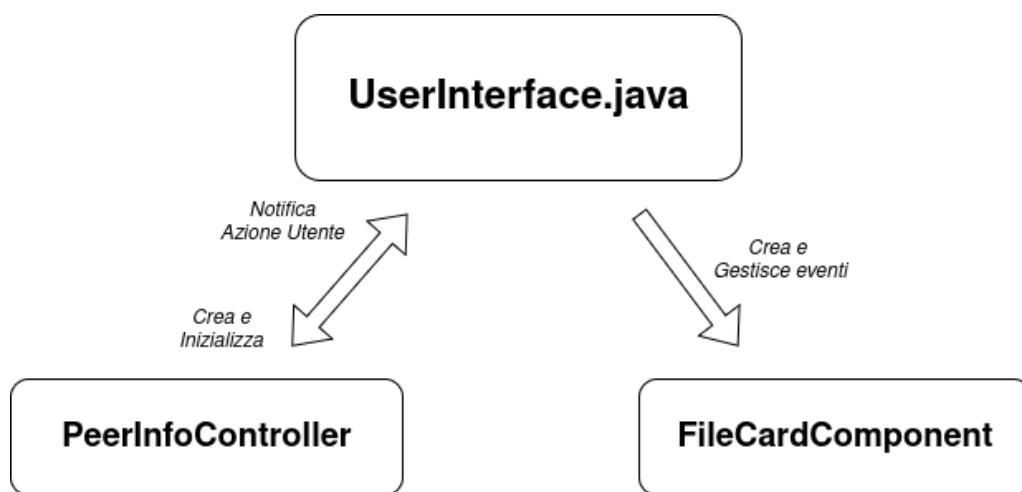


Figura 3.10: Architettura a componenti del Controller della UI.

Capitolo 4

Implementazione delle difese in rete

4.1 Introduzione e obiettivi del componente NGFW cloud

Sottoporre il traffico generato dai client a un'attenta analisi è l'unico modo per applicare le strategie proattive che coniugano i benefici delle difese locali con quelli delle difese applicate a livello dell'infrastruttura di rete. Per questo motivo negli anni sono state sviluppate dalle aziende di cyber-sicurezza — come Fortinet, Palo Alto e Cisco — soluzioni aggregate che potessero integrare tutte le strategie di sicurezza prima delegate ad apparati singoli, raggruppandole in un unico componente.

Da questa visione derivano i moderni Next Generation Firewall (NGFW), ovvero componenti hardware e software di livello tipicamente enterprise, fondamentali per l'integrazione completa delle difese in modo semplice e centralizzato (Figura 4.1).

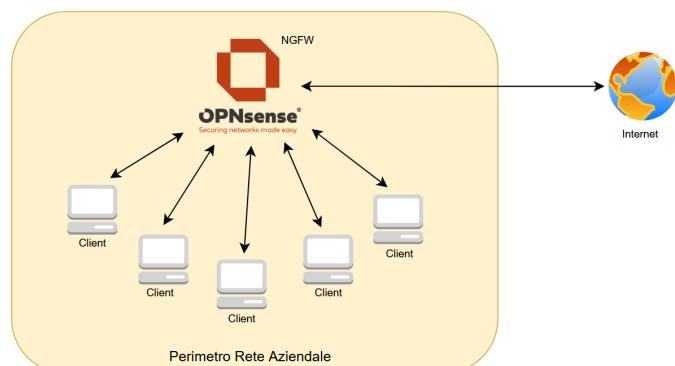


Figura 4.1: Tipica architettura di rete con un NGFW.

Per quanto questi strumenti siano diffusi e distribuiti da aziende leader, non è presente una grande abbondanza di soluzioni open-source. Per questo progetto, che vogliamo sia il più riproducibile e svincolato possibile, abbiamo scelto di utilizzare OPNSense: una soluzione NGFW open-source basata su FreeBSD e derivata dal più famoso pfSense, che integra al suo interno tutte le funzionalità necessarie.

Nel panorama odierno, questi strumenti vengono distribuiti su macchine (server) preconfigurate, pronte per essere installate all'interno delle reti aziendali senza modificarne la struttura.

Questo progetto si propone invece di implementare un NGFW non all'interno di una rete fisica aziendale, ma su un server cloud, in modo da coniugare la semplicità di gestione e la scalabilità tipiche delle soluzioni cloud con le funzionalità avanzate di un NGFW (Figura 4.2).

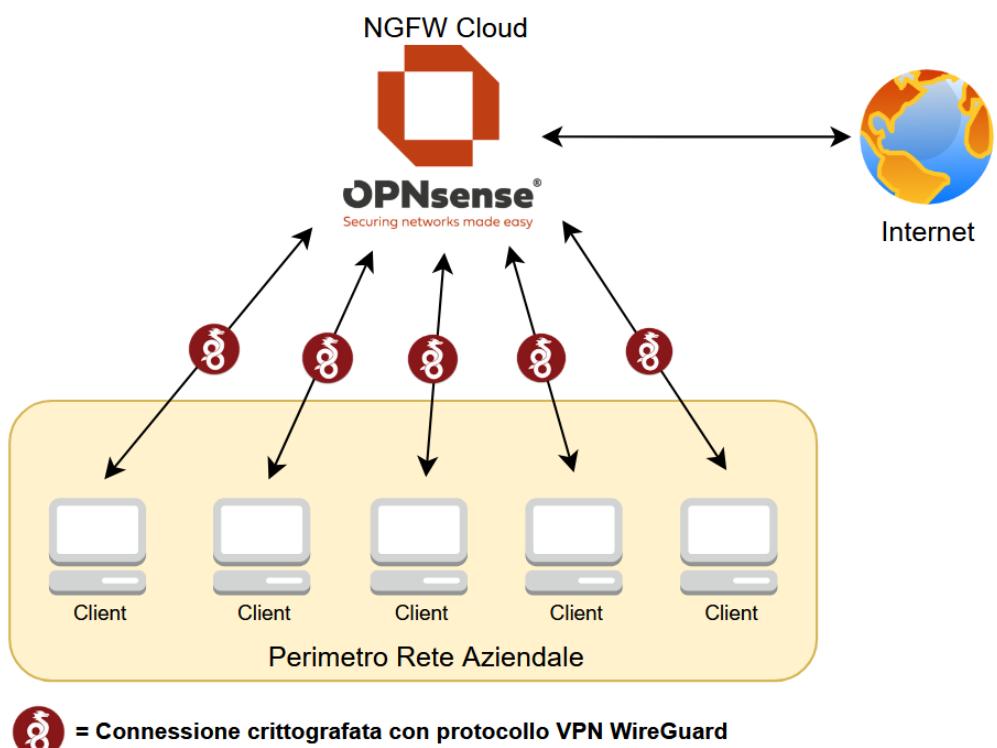


Figura 4.2: Schema logico dell'architettura di rete che si vuole realizzare.

Seppure i NGFW siano pensati per essere installati in difesa di risorse esposte pubblicamente da minacce provenienti dall'esterno della rete — come un sito web o servizi aziendali — in questo progetto si vuole dimostrare come questi possano essere utilizzati anche per la protezione esclusiva e mirata sui singoli client all'interno della rete, garantendone inoltre l'isolamento e la sicurezza in modo slegato dal perimetro fisico della rete aziendale.

Per esempio, un client che si trova a Zurigo(CH), instradando tutto il traffico tramite la VPN a un NGFW ospitato a Francoforte (DE), sarebbe comunque soggetto alle regole di sicurezza e alle strategie di difesa che esso applica (Figura 4.3).



Figura 4.3: Schema logico di comunicazione Client-NGFW-Internet.

Il sistema prevede l'integrazione del NGFW cloud direttamente nel percorso del traffico di rete, permettendo così la completa scansione del traffico con l'obiettivo di individuare pattern malevoli, bloccare le connessioni non autorizzate e filtrare le richieste DNS. Le strategie di difesa introdotte risultano così completamente indipendenti dall'infrastruttura fisica, che diventa un semplice mezzo di trasporto, insicuro e neutrale rispetto al contenuto informativo dei pacchetti che vi transitano.

4.2 Configurazione dell'infrastruttura di base dell'NGFW

La realizzazione dell'infrastruttura implica diversi aspetti, che vanno dalla scelta del cloud provider, alla progettazione dell'infrastruttura e alla configurazione della Virtual Machine (VM). In questa sezione, verranno descritte le scelte tecniche adottate per la messa in opera dell'NGFW, assieme agli schemi finali che ne descrivono il funzionamento e il corretto flusso dei pacchetti.

4.2.1 Scelta del cloud provider e progettazione dell'infrastruttura

La scelta del cloud provider costituisce una decisione strategica di rilevanza fondamentale, in quanto da essa dipende l'implementazione e la gestione dell'intera architettura di sicurezza. Tra i principali provider del settore — tra cui AWS e Microsoft Azure — per le finalità della presente tesi è stato selezionato Oracle Cloud Infrastructure (OCI). Questa decisione non è motivata da vincoli di sicurezza (le cui informazioni a riguardo sono spesso omesse) o di disponibilità delle risorse (considerate illimitate per filosofia cloud), ma dalla possibilità di utilizzare un'istanza gratuita con risorse adeguate e dall'alta possibilità di personalizzazione che un provider cloud strutturato e corporate come OCI può offrire (Figura 4.4). Va tuttavia sottolineato che gli aspetti di sicurezza dell'infrastruttura e di disponibilità delle risorse sono da valutare con attenzione se si volesse implementare questo tipo di soluzione per ambienti critici o aziendali.

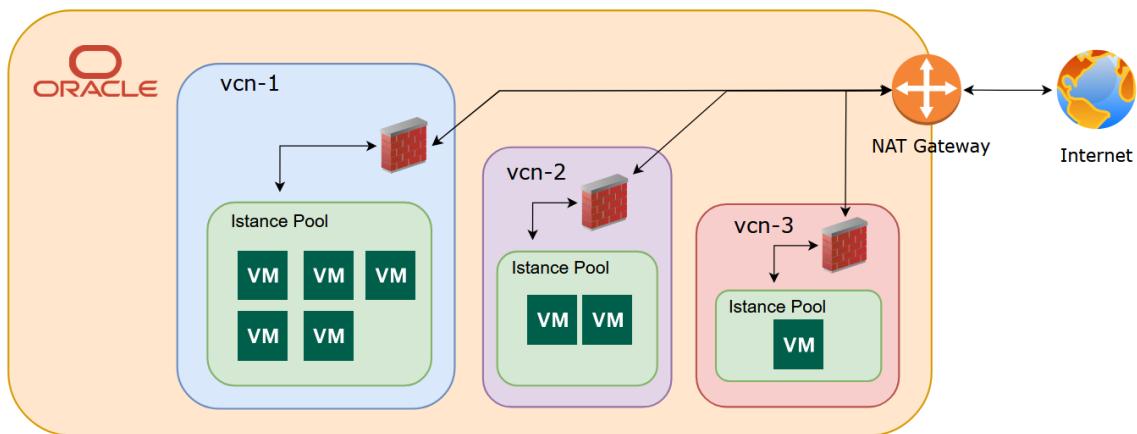


Figura 4.4: Schema architetturale di riferimento in OCI [30].

Considerando lo stato attuale del progetto e alla leggerezza del sistema NGFW adottato, i requisiti infrastrutturali risultano limitati all'utilizzo di una singola macchina virtuale, collocata all'interno di una rete dedicata. Come avviene per ogni risorsa di calcolo in Oracle Cloud Infrastructure, tale rete è soggetta, per motivi di sicurezza e di gestione, alle regole di traffico definite dalla corrispondente Virtual Cloud Network (VCN) [31]. Inoltre, per garantire la connettività con i client VPN, essa è

interconnessa ad una rete pubblica, la quale abilita l'accesso ad internet e lo scambio di dati con l'esterno (Figura 4.5).

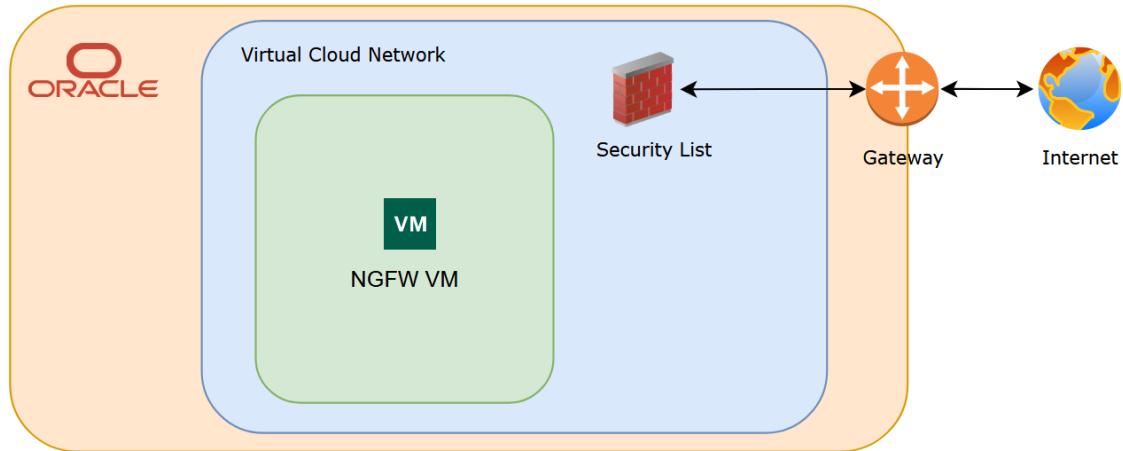


Figura 4.5: Architettura semplificata implementata su OCI per il progetto [30].

Data la limitata disponibilità di VM x86 e x64 nel piano gratuito di OCI, per l'installazione di OPNSense è stata utilizzata una versione modificata e ricompilata compatibile con CPU ARM, realizzata da *Maurice Walker* e completamente open source [32]. Questa scelta è stata obbligatoria per poter eseguire OPNSense — compatibile ufficialmente solo per sistemi x86 e x64 — in macchine virtualizzate su architetture aarch64, come le VM.Standard.A1.Flex di Oracle Cloud Infrastructure.

4.2.2 Configurazione Security List relativa alla VCN

Ogni VCN in Oracle Cloud Infrastructure ha una *Security List* associata, che permette di definire le regole di traffico in ingresso e in uscita per le risorse della rete; queste regole di traffico sono applicate a tutte le risorse assegnate alla VCN, e di conseguenza anche alle eventuali Virtual Machine.

Ciò comporta che per poter accedere alla VM che ospita OPNSense, sia necessario impostare precise regole di traffico in ingresso e in uscita, se non ancora configurate.

Ingress Rules

Ingress Rules							
	Source	IP Protocol	Source Port Range	Destination Port Range	Type and Code	Allows	Description
<input type="checkbox"/>	No  /32	TCP	All	22	TCP traffic for ports: 22 SSH Remote Login Protocol		...
<input type="checkbox"/>	0.0.0.0/0	UDP	All	51820	UDP traffic for ports: 51820		...
<input type="checkbox"/>	::/0	UDP	All	51820	UDP traffic for ports: 51820		...
<input type="checkbox"/>	No  /32	TCP	All	22	TCP traffic for ports: 22 SSH Remote Login Protocol	SSH remote	...
<input type="checkbox"/>	0.0.0.0/0	TCP	All	80	TCP traffic for ports: 80	HTTP	...
<input type="checkbox"/>	0.0.0.0/0	TCP	All	443	TCP traffic for ports: 443 HTTPS	HTTPS	...
<input type="checkbox"/>	10.0.0.0/24	ICMP		All	ICMP traffic for: All		...

Page 1 of 1 (1 - 7 of total items) Items per page 25

Figura 4.6: Regole di ingresso - Security list, VCN.

In ingresso, sono state definite le regole per permettere l'accesso SSH, HTTP-HTTPS e protocollo WireGuard (VPN) (Figura 4.6), mentre sono state lasciate invariate le regole di gestione del traffico in uscita, che sono impostate di default per permettere le comunicazioni verso internet (Figura 4.7).

Egress Rules

Egress Rules							
	Destination	IP Protocol	Source Port Range	Destination Port Range	Type and Code	Allows	Description
<input type="checkbox"/>	No 0.0.0.0/0	All Protocols				All traffic for all ports	...
<input type="checkbox"/>	::/0	All Protocols				All traffic for all ports	...

Page 1 of 1 (1 - 2 of total items) Items per page 25

Figura 4.7: Regole di uscita - Security list, VCN.

Le regole configurate consentono la connessione alla macchina virtuale che ospita OPNsense, al fine di permetterne la configurazione e la gestione previa autenticazione. Inoltre, le regole di ingresso sulla porta 51820 consentono ai client di instaurare connessioni con il daemon WireGuard installato sul NGFW, garantendo l'accesso alla rete posta sotto protezione.

4.3 Configurazione del server VPN per l'accesso client

Per permettere l'accesso remoto da parte dei client al NGFW, è necessario configurare un daemon VPN al quale essi possano connettersi; tra le varie soluzioni disponibili, si può optare per OpenVPN, che offre una configurazione molto sicura ma più pesante, oppure per *WireGuard* [33], noto per la sua leggerezza, sicurezza ed efficienza.

Uno dei motivi che ha guidato la scelta di WireGuard è la sua piena integrazione all'interno di OPNSense; questa compatibilità nativa consente una configurazione rapida e diretta dell'interfaccia VPN, evitando la necessità di installare pacchetti esterni o gestire manualmente le configurazioni (Figura 4.8).

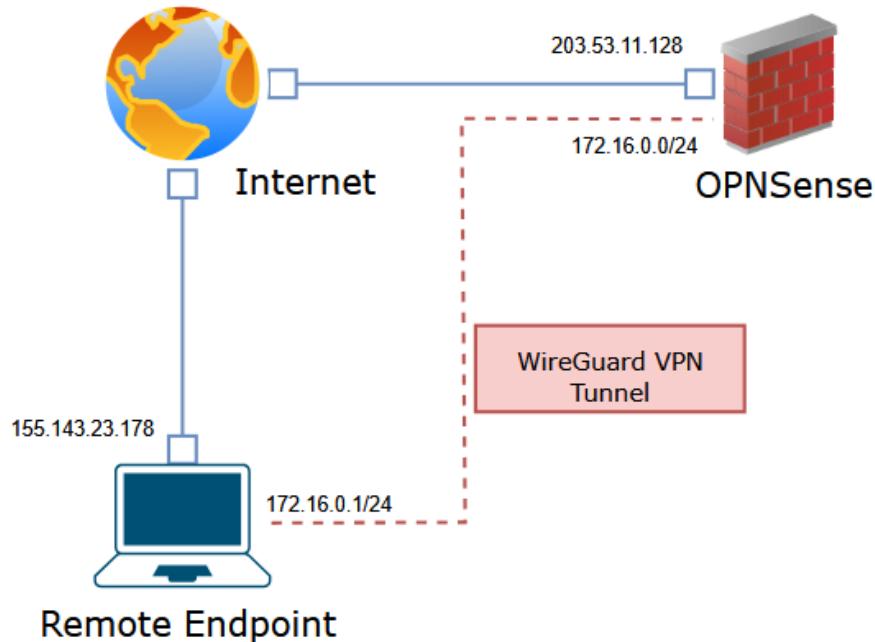


Figura 4.8: Schema di rete - connessione tra endpoint e OPNSense tramite tunnel WireGuard.

Dal punto di vista della disponibilità, WireGuard è un protocollo open-source e multipiattaforma, disponibile per i principali sistemi operativi — quali Windows, macOS, Linux (cli-only), iOS e Android — grazie al client ufficiale. Sebbene minimale, il client offre comunque alcune funzionalità fondamentali per garantire la sicurezza dell'utente, tra cui la possibilità di forzare il reindirizzamento del traffico

di rete attraverso il tunnel VPN (nota come funzionalità di *Kill-Switch*), la quale impedisce che dati sensibili vengano trasmessi all'esterno del tunnel in caso di disconnessione.

4.3.1 Funzionamento e configurazione di WireGuard

Similmente ad altri protocolli crittografici – come HTTPS – WireGuard si fonda su un meccanismo di scambio e riconoscimento di chiavi pubbliche. Ogni peer (sia esso client o server) conosce la chiave pubblica dell'altro peer e la associa a un'interfaccia di rete virtuale dotata di un indirizzo IP dedicato. Tale interfaccia consente ai due dispositivi di comunicare in modo sicuro e autenticato (Figura 4.9).

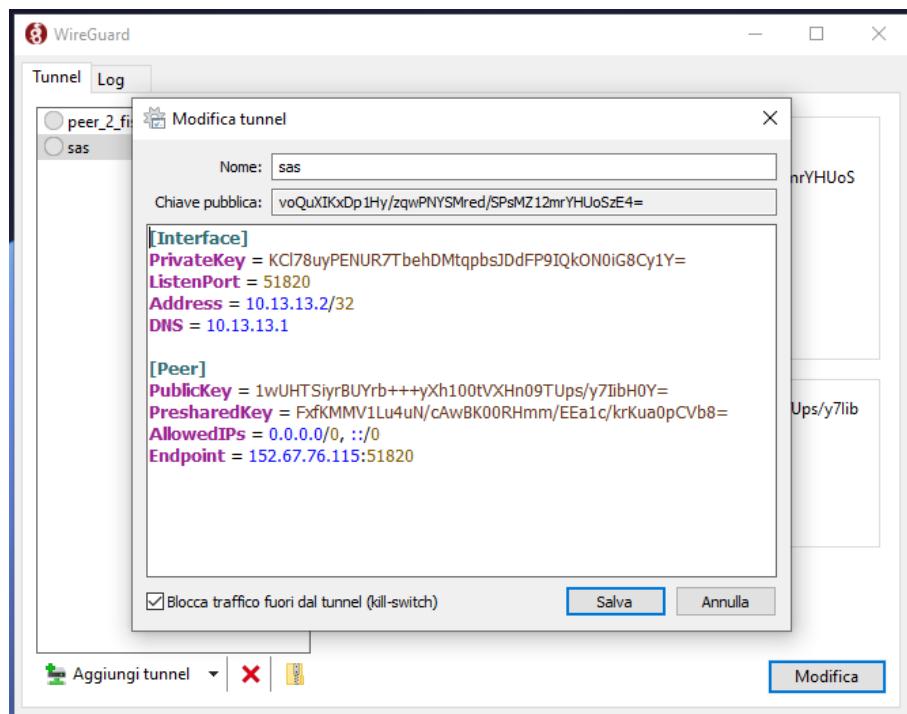


Figura 4.9: Esempio di configurazione WireGuard.

In particolare, ogni peer viene configurato specificando:

- La propria coppia di chiavi: una chiave privata, mantenuta segreta, e la corrispondente chiave pubblica, condivisa con gli altri peer;
- L'endpoint di riferimento (indirizzo IP e porta), necessario quando un peer deve essere raggiunto attivamente dall'altro peer;

- La lista delle reti consentite (**AllowedIPs**), che determina quali pacchetti IP devono essere instradati attraverso quel peer. In questo modo si definisce sia la destinazione del traffico sia la funzione di routing all'interno del tunnel.

La comunicazione può quindi iniziare: il client invia pacchetti al server (il cui indirizzo e chiave pubblica sono già noti), e il server li accetta perché riconosce la chiave pubblica del client nella propria configurazione. Instradando il traffico attraverso l'interfaccia virtuale di WireGuard, tutto il flusso dati viene convogliato verso il NG-FW. Su quest'ultimo è quindi necessario configurare una corrispondente interfaccia virtuale dedicata (Figura 4.10), che permette al firewall di applicare le proprie regole di sicurezza e di gestire il traffico cifrato proveniente dai client VPN.



Figura 4.10: Interfaccia OPNSense assegnata a WireGuard.

La quale verrà utilizzata per applicare le regole di traffico dal firewall e dagli altri applicativi di rete.

4.4 Applicazione delle regole firewall personalizzate

Per applicare delle regole efficienti, bisogna prima valutare il tipo di traffico che dovranno andare a filtrare, in modo da garantire un adeguato livello di protezione senza causare disservizi; per far ciò, bisogna conoscere molto bene il tipo di applicazioni utilizzate dall'utente e valutare quali limitazioni gli possono essere imposte: se un utente fa uso frequente di SSH, per quanto questo protocollo venga utilizzato da una nicchia di utenza, non potremo bloccarlo.

Su questa filosofia possiamo analizzare il nostro obiettivo, ovvero la protezione dei dispositivi client: questi dispositivi non espongono tipicamente dei socket e quindi non hanno la necessità di essere protetti da connessioni derivanti dall'esterno della rete, tuttavia se infettati, potrebbero voler comunicare con server di Comando e

Controllo, e in questo caso il blocco delle porte verso l'esterno potrebbe risultare efficace.

Potremmo quindi decidere di bloccare tutte le comunicazioni TCP/UDP provenienti dai client, permettendo tuttavia il passaggio di quei pacchetti che soddisfano alcune specifiche regole che consideriamo sicure (Figura 4.11):

- 80/TCP: Per permettere il traffico HTTP;
- 443/TCP e 443/UDP: Per permettere il traffico HTTPS e QUIC;
- 143/TCP: Per permettere il traffico IMAP;
- 993/TCP: Per permettere il traffico IMAP sicuro (IMAPS/STARTTLS);
- 110/TCP: Per permettere il traffico POP3;
- 995/TCP: Per permettere il traffico POP3 sicuro (POP3S/STARTTLS);
- 25/TCP: Per permettere il traffico SMTP;
- 465/TCP: Per permettere il traffico SMTP sicuro (SMTPS/STARTTLS);
- 22/TCP: Per permettere il traffico SSH;
- 50000–65535/UDP: Per permettere il traffico UDP a Discord;

<input type="checkbox"/>		IPv4+6 TCP	Tunnel0 net	*	*	80 (HTTP)	*	*	Allow TCP HTTP				
<input type="checkbox"/>		IPv4+6 TCP	Tunnel0 net	*	*	443 (HTTPS)	*	*	Allow TCP HTTPS				
<input type="checkbox"/>		IPv4+6 UDP	Tunnel0 net	*	*	443 (HTTPS)	*	*	Allow UDP QUIC HTTPS				
<input type="checkbox"/>		IPv4+6 TCP	Tunnel0 net	*	*	143 (IMAP)	*	*	Allow TCP IMAP				
<input type="checkbox"/>		IPv4+6 TCP	Tunnel0 net	*	*	993 (IMAP/S)	*	*	Allow TCP IMAP STARTTLS				
<input type="checkbox"/>		IPv4+6 TCP	Tunnel0 net	*	*	110 (POP3)	*	*	Allow TCP POP3				
<input type="checkbox"/>		IPv4+6 TCP	Tunnel0 net	*	*	995 (POP3/S)	*	*	Allow TCP POP3 STARTTLS				
<input type="checkbox"/>		IPv4+6 TCP	Tunnel0 net	*	*	25 (SMTP)	*	*	Allow TCP SMTP				
<input type="checkbox"/>		IPv4+6 TCP	Tunnel0 net	*	*	465 (SMTP/S)	*	*	Allow TCP SMTP STARTTLS				
<input type="checkbox"/>		IPv4+6 TCP	Tunnel0 net	*	*	22 (SSH)	*	*	Allow TCP SSH				
<input type="checkbox"/>		IPv4+6 UDP	Tunnel0 net	*	*	50000 - 65535	*	*	Allow UDP Discord				
<input type="checkbox"/>		IPv4+6 TCP/UDP	Tunnel0 net	*	*	*	*	*	Block outbound TCP/UDP				

Figura 4.11: Configurazione firewall per il blocco di tutti i pacchetti che non corrispondono con le esclusioni.

Questa configurazione, seppure restrittiva, permette di bloccare la maggior parte delle comunicazioni non autorizzate, mitigando almeno in parte due importanti problematiche di sicurezza di seguito riportate.

4.4.1 Isolamento dei client connessi al NGFW

Un’eventuale infezione da malware potrebbe portare alla compromissione di altri dispositivi connessi alla stessa rete: sfruttando la rete locale di WireGuard come vettore di diffusione esso potrebbe diffondersi lateralmente su tutti i client connessi al NGFW. Per prevenire questo scenario, è stata introdotta una regola che blocca tutte le comunicazioni tra i client connessi al NGFW, de facto permettendone l’isolamento reciproco e limitando così la diffusione di un eventuale malware.

4.4.2 Protezione contro DNS leak e DNS-over-HTTPS (DoH)

La risoluzione DNS non è una pratica esclusiva del sistema operativo: alcuni virus avanzati possono integrare un proprio resolver DNS, configurato per raggiungere server pubblici a cui far risolvere i propri domini malevoli, bypassando il blocco risolutivo da noi applicato ed esponendo il dispositivo a possibili minacce. Per bloccare ai client questa possibilità, è necessario bloccare ogni tipo di traffico UDP verso internet che abbia la 53 come porta di destinazione, garantendo però che il traffico DNS diretto verso il resolver ricorsivo locale del NGFW non venga alterato (Figura 4.12).

<input type="checkbox"/>					IPv4 UDP	Tunnel0 net	*	172.16.0.1	53 [DNS]	*	*	Allow packets to local resolver				
<input type="checkbox"/>					IPv4+6 *	Tunnel0 net	*	Tunnel0 net	*	*	*	Block intra-peer communications				
<input type="checkbox"/>					IPv4+6 UDP	Tunnel0 net	*	*	53 [DNS]	*	*	Block all DNS packets outbound				

Figura 4.12: Configurazione firewall per il blocco di tutti i pacchetti UDP con destinazione diversa da 172.16.0.1 (Resolver DNS Caching nella sottorete wg0).

Un altro rischio sorge nell'utilizzo da parte di alcuni malware avanzati del protocollo DNS-over-HTTPS (DoH), che sfrutta l'incapsulamento HTTPS per il trasporto di richieste DNS, permettendo così di sfruttare server esterni — come quelli di Google, Cloudflare, o Quad9 — rendendoci impossibile il blocco e filtraggio delle richieste DNS. Questa tecnica di evasione permette ad eventuali malware di risolvere indirizzi IP malevoli direttamente dai server pubblici attraverso chiamate HTTPS, bypassando completamente i controlli DNS imposti dal NGFW e riuscendo così a risolvere gli IP dei propri server di Comando e Controllo (se non presenti nelle regole IPS).

Per mitigare questa vulnerabilità, è stata creata una serie di regole firewall che bloccano attivamente l'accesso a tutti i server DoH noti, indipendentemente dalla porta e dal protocollo utilizzati per la comunicazione (Figura 4.13).



Figura 4.13: Configurazione firewall per il blocco di tutti i pacchetti destinati a server DoH.

Ai fini di questo progetto è stata utilizzata una lista di server DoH aperta e aggiornata quotidianamente; per garantire inoltre che le modifiche venissero recuperate ed applicate dal NGFW, è stato possibile configurare OPNSense per il pull automatico dalla repository ufficiale della lista (Figura 4.14).

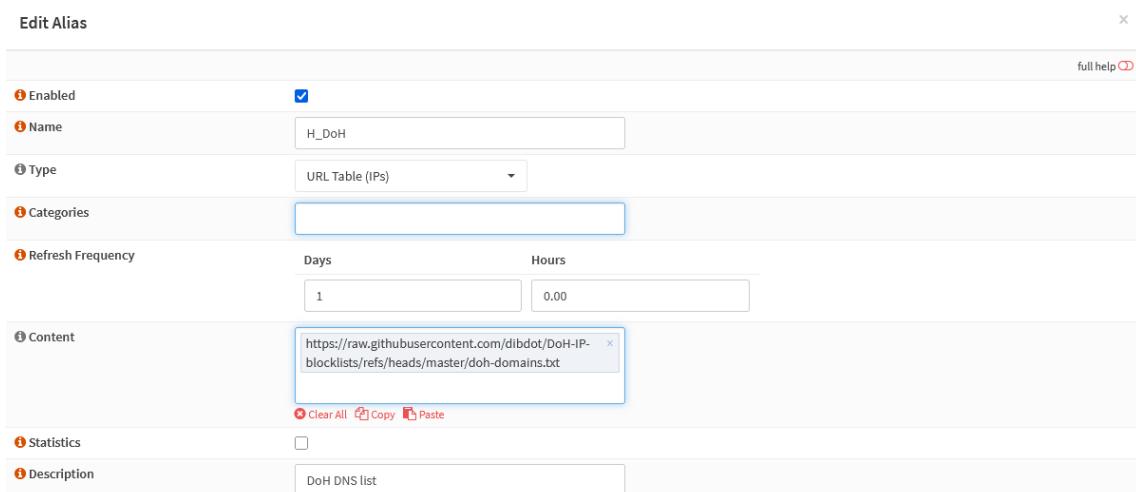


Figura 4.14: Configurazione dell'alias per la lista dei server DoH utilizzata nel firewall.

Nonostante tutte le precauzioni prese, va considerato che qualsiasi strategia difensiva potrebbe essere alla lunga aggirata: infatti, nessuno vieta a un'organizzazione criminale di creare di volta in volta nuovi server DoH ricorsivi, dismettendo quelli individuati, tantomeno, nessuno vieta a un server C2 di essere ospitato sulla porta 80 o 443. Tuttavia nella corsa contro il cybercrimine ogni layer di sicurezza aggiuntivo può fare la differenza, costringendo i criminali a intraprendere strategie dispendiose e permettendo ai sistemi di sicurezza di individuare una minaccia in tempo.

4.5 Configurazione componente IPS/IDS

IPS e IDS sono sistemi fondamentali per la prevenzione degli incidenti informatici, essi sono in grado di identificare pattern malevoli e di applicare regole di sicurezza in maniera tempestiva, prevenendo così eventuali violazioni. OPNSense implementa queste funzionalità mediante *Suricata* [34], un software open-source ad alte prestazioni per l'analisi delle reti e il rilevamento delle minacce, installato di default e configurabile tramite diverse liste di regole pubbliche (Figura 4.15).

The screenshot shows the 'Rulesets' tab of the OPNSense configuration interface. At the top, there are tabs for 'Settings', 'Download', 'Rules', 'User defined', 'Alerts', and 'Schedule'. Below the tabs, there's a section for 'Rulesets' with buttons for 'Enable selected' and 'Disable selected'. A search bar is also present. The main area displays a table of available rules, with columns for 'Description', 'Last updated', 'Enabled', and 'Edit'. The table lists various sources like abuse.ch and ET open, with their last update times and current enabled status.

Description	Last updated	Enabled	Edit
abuse.ch/Feodo Tracker	not installed	x	
abuse.ch/SSL Fingerprint Blacklist	not installed	x	
abuse.ch/SSL IP Blacklist	not installed	x	
abuse.ch/ThreatFox	not installed	x	
abuse.ch/URLhaus	2025/07/16 2:00	✓	
ET open/3coresec	2025/07/16 2:00	✓	
ET open/botcc	not installed	x	
ET open/botcc.portgrouped	not installed	x	
ET open/ciarmy	2025/07/16 2:00	✓	
ET open/compromised	2025/07/16 2:00	✓	
ET open/drop	2025/07/16 2:00	✓	

Download & Update Rules

Figura 4.15: Liste di regole importatibili e disponibili di default.

La scelta delle liste da utilizzare è un passo cruciale e va valutato sulla base del tipo di protezione che si sta cercando, infatti, una lista focalizzata sulla prevenzione delle SQL Injection potrebbe risultare utile per un server web, ma del tutto irrilevante nel nostro contesto d'uso; per questo motivo è ancora una volta essenziale conoscere molto bene il tipo di traffico che genereranno i client prima di prendere decisioni.

La maggior parte delle liste disponibili viene aggiornata con cadenza oraria, e sebbene non sia possibile analizzarle tutte nel dettaglio, offrono spesso indicatori di qualità sufficienti per contrastare minacce generiche su larga scala. L'aggiornamento di queste liste è affidato a bot dedicati, che monitorano costantemente lo spazio IPv4/IPv6 pubblico a livello globale, alla ricerca di server di Comando e Controllo (C2), malware, siti di phishing e altre minacce.

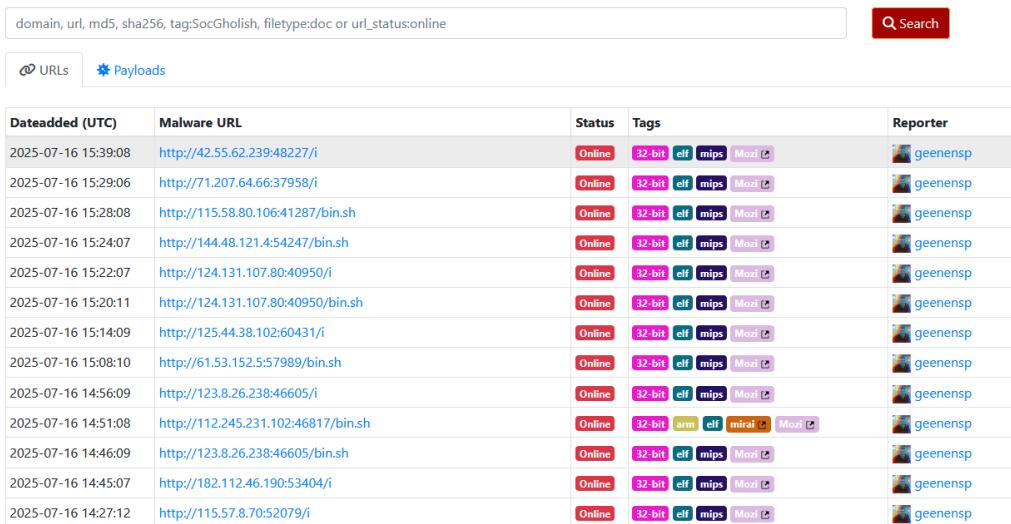
Possiamo di seguito elencare quelle da noi implementate:

- **Emerging Threats (ET)**: Disponibili sia in versione open source (*ET Open*) che commerciale (*ET Pro*), forniscono regole aggiornate e specifiche per categoria di minaccia; presentano una struttura granulare, che nel nostro caso di studio ha permesso di implementare le seguenti liste specifiche:

- 3CoreSec
- CIArmy
- Compromised
- Drop
- Emerging Phishing
- Emerging Web_Client
- Threatview_CS_c2

Selezionate per via della loro pertinenza nel contrasto delle minacce verso i client (*ET Category Descriptions*, [35])

- **Abuse.ch rulesets**: Progetti come SSLBL, ThreatFox e URLhaus forniscono regole mirate contro minacce specifiche, tra cui malware, server C2 e domini sospetti. In particolare, nel caso di studio analizzato, la lista URLhaus (Figura 4.16, Figura 4.17) si rivela particolarmente rilevante, poiché è progettata per bloccare la comunicazione verso domini, host e URL malevoli.



The screenshot shows a web-based interface for URLhaus. At the top, there is a search bar with placeholder text: "domain, url, md5, sha256, tag:SocGholish, filetype:doc or url_status:online". To the right of the search bar is a red "Search" button. Below the search bar, there are two tabs: "URLs" (selected) and "Payloads". The main area is a table with the following columns: Dateadded (UTC), Malware URL, Status, Tags, and Reporter. The table lists 15 rows of data, each corresponding to a different URL added on July 16, 2025, between 15:39:08 and 14:27:12 UTC. The URLs are mostly .bin.sh files from various IP addresses. The "Status" column shows "Online" for all entries. The "Tags" column includes "32-bit", "elf", "mips", and "Mozi". The "Reporter" column consistently shows "geenensp".

Dateadded (UTC)	Malware URL	Status	Tags	Reporter
2025-07-16 15:39:08	http://42.55.62.239:48227/i	Online	32-bit elf mips Mozi	geenensp
2025-07-16 15:29:06	http://71.207.64.66:37958/i	Online	32-bit elf mips Mozi	geenensp
2025-07-16 15:28:08	http://115.58.80.106:41287/bin.sh	Online	32-bit elf mips Mozi	geenensp
2025-07-16 15:24:07	http://144.48.121.4:54247/bin.sh	Online	32-bit elf mips Mozi	geenensp
2025-07-16 15:22:07	http://124.131.107.80:40950/i	Online	32-bit elf mips Mozi	geenensp
2025-07-16 15:20:11	http://124.131.107.80:40950/bin.sh	Online	32-bit elf mips Mozi	geenensp
2025-07-16 15:14:09	http://125.44.38.102:60431/i	Online	32-bit elf mips Mozi	geenensp
2025-07-16 15:08:10	http://61.53.152.5:57989/bin.sh	Online	32-bit elf mips Mozi	geenensp
2025-07-16 14:56:09	http://123.8.26.238:46605/i	Online	32-bit elf mips Mozi	geenensp
2025-07-16 14:51:08	http://112.245.231.102:46817/bin.sh	Online	32-bit arm elf mirai Mozi	geenensp
2025-07-16 14:46:09	http://123.8.26.238:46605/bin.sh	Online	32-bit elf mips Mozi	geenensp
2025-07-16 14:45:07	http://182.112.46.190:53404/i	Online	32-bit elf mips Mozi	geenensp
2025-07-16 14:27:12	http://115.57.8.70:52079/i	Online	32-bit elf mips Mozi	geenensp

Figura 4.16: Lista URLhaus (URL) via WebGUI.

Firstseen (UTC)	SHA256 hash	File type	Bazaar	Signature
2025-07-16 15:39:22	c4d628261bf6c9de72822df555efaf24c7715549897d5b03ca5eee6d6413e7d5	html		
2025-07-16 15:39:05	f95d32f4df5395323a7a3e0f3826981f040a96652680b5e4ee5d18aed3286bad	html		
2025-07-16 15:29:33	23f08fc590def51e49cd04c7a6f385eec3d7be62b631dc85b87d4b0714d53993	zip		
2025-07-16 15:24:38	29a7b88add7fb07326791ecfb0b26e081bdd5c9e839ba7462071e7ea7c50efa9	html		
2025-07-16 15:12:17	10d874ae700a41bcc4d4acac8bb29cbc6661496ea170e1ada7d84a16e8a62f51	js		
2025-07-16 15:10:46	ab049a916021a60c7e144cbf855dfda63e239e1dafaace636482f9c74004d3fb	elf		Mirai
2025-07-16 14:47:57	90c99af8a53605a8e41c7b13a27c98c14fb0dd1619474e246f5b0e1340092ab5	unknown		
2025-07-16 13:57:11	8c95aa59c329808aa116cc779d4ae024673947862713a2417c15b289349d61c2	msi		RustyStealer
2025-07-16 13:57:11	c89111d126874c34430e868550615765c0d623a517dd3ceb4b60e8a188244462	exe		BlankGrabber
2025-07-16 13:57:08	f0b1373001ff7acf13a034096aba156d9c691c5690a9568a67b216504c3d4766	msi		RustyStealer
2025-07-16 13:57:08	4ca019bbb409f45e4f14d1b80fea6b3ef6248a7ea26ffffd592f360b81e7d664	exe		
2025-07-16 13:24:48	270b0a8282f042b4f7ab0607ac90ccf577aff14e415cd645ef1eb909a3bf92ad	html		
2025-07-16 13:11:26	335ad3f112ab84d597a4767a840e612f63958f36d723cd32f4522c963d28eb06	unknown		
2025-07-16 12:55:08	4e487b7d580b05ea9a89866843c834b7ad6b010dff5e8536491bfe0cc28e9478	js		
2025-07-16 12:48:42	40015120b29d50b1287dc54951e13ae5e2aeb9930d045a4d4f5a4e6af6ad3c9	unknown		

Figura 4.17: Lista URLhaus (Payloads) via WebGUI.

4.6 Configurazione del sistema di filtraggio DNS e DNS Security

La risoluzione controllata dei nomi di dominio rappresenta una strategia estremamente potente, ma al tempo stesso un tema particolarmente delicato; un risolutore DNS ricorsivo consente, infatti, di applicare delle blocklist per limitare la navigazione, registrare tutti i domini a cui un determinato client ha tentato di accedere, e allo stesso tempo, sfruttare meccanismi di caching per velocizzare il processo di risoluzione. Per questo scopo, OPNSense integra il resolver ricorsivo e di caching *Unbound* [36], software open-source, che configurato con adeguate blocklist, permette di filtrare la risoluzione dei domini impedendo l'accesso a contenuti malevoli come phishing, malware o adware.

Per comprendere appieno il funzionamento di un resolver come *Unbound*, illustriamo di seguito gli step che esegue ogni qualvolta un client inoltra una richiesta:

- Richiesta DNS dal client:** il client invia una query DNS ad Unbound per risolvere il dominio `malicious-domain.com`.
- Verifica della cache di Unbound:** Unbound controlla la propria cache per verificare se ha già una risposta valida per quella query. Se presente, la risposta viene inviata immediatamente al client, riducendo così i tempi di risposta.
- Controllo della blacklist:** se la risposta non è presente nella cache, Unbound confronta il dominio richiesto con le voci presenti nella blocklist configurata.
- Risposta modificata:** se il dominio è presente nella blocklist, Unbound risponde al client con un indirizzo IP nullo (`0.0.0.0`).
- Risoluzione normale:** se il dominio non è presente nella blocklist, Unbound procede con la risoluzione ricorsiva, interrogando i server DNS di tipo root, TLD e infine Autoritativi del dominio richiesto (Figura 4.18).

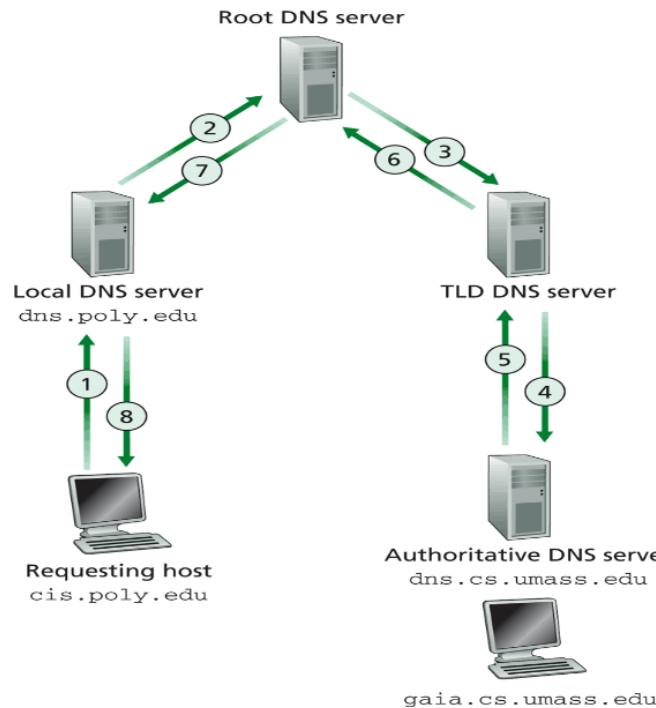


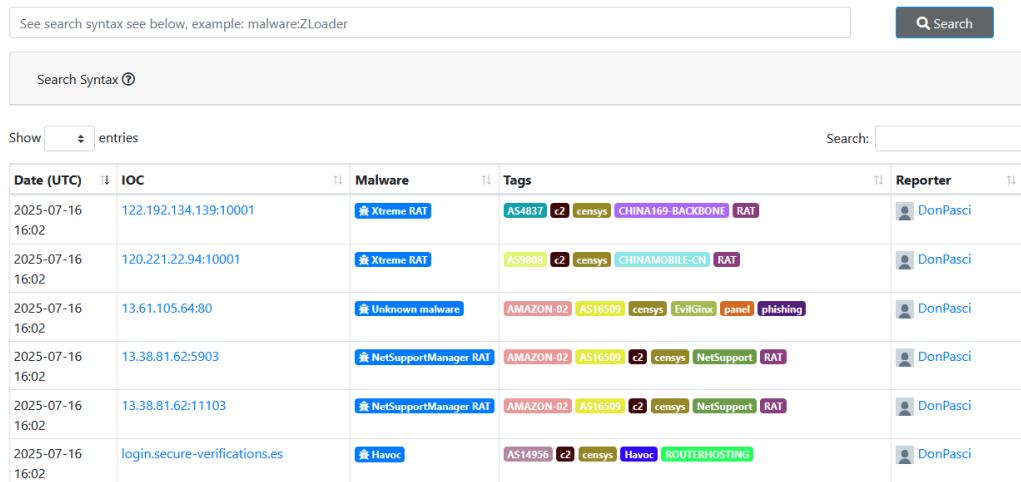
Figura 4.18: Flusso base per la risoluzione DNS [37].

- Caching della risposta:** la risposta ottenuta, sia essa bloccata o valida, viene memorizzata nella cache di Unbound, riducendo il carico di lavoro per richieste successive allo stesso dominio.

Un ruolo importante lo svolge quindi anche il root DNS server, che può contribuire alla protezione bloccando a monte la risoluzione di un dominio, oppure può mettere in atto procedure di tracciamento per minare la privacy degli utenti; è quindi sempre un bene scegliere con cura questi strumenti, affidandosi a soluzioni il più trasparenti possibile. Nel contesto di questo progetto è stato scelto `dns0.eu`, un provider DNS europeo che mette a disposizione resolver *zero-trust*, progettati specificamente per ambienti ad alta sensibilità e orientati alla massima tutela della privacy.

A complemento di questa scelta, sono state configurate localmente delle blocklist mirate, accuratamente scelte per garantire un filtraggio pertinente ed efficace.

- **Abuse.ch – ThreatFox:** una lista basata su un ampio database di Indicatori di Compromissione (IOC), utile per il blocco di domini associati a malware, C2 e altre minacce (Figura 4.19).



The screenshot shows the ThreatFox web interface. At the top, there is a search bar with placeholder text "See search syntax see below, example: malware:ZLoader" and a "Search" button. Below the search bar is a "Search Syntax" section with a help icon. The main area displays a table of IOC entries. The table has columns: Date (UTC), IOC, Malware, Tags, and Reporter. The table contains the following data:

Date (UTC)	IOC	Malware	Tags	Reporter
2025-07-16 16:02	122.192.134.139:10001	Xtreme RAT	AS4837 c2 censys CHINA169-BACKBONE RAT	DonPasci
2025-07-16 16:02	120.221.22.94:10001	Xtreme RAT	AS9808 c2 censys CHINAMOBILE-CN RAT	DonPasci
2025-07-16 16:02	13.61.105.64:80	Unknown malware	AMAZON-02 AS16509 censys EvilGinx panel phishing	DonPasci
2025-07-16 16:02	13.38.81.62:5903	NetSupportManager RAT	AMAZON-02 AS16509 c2 censys NetSupport RAT	DonPasci
2025-07-16 16:02	13.38.81.62:11103	NetSupportManager RAT	AMAZON-02 AS16509 c2 censys NetSupport RAT	DonPasci
2025-07-16 16:02	login.secure-verifications.es	Havoc	AS14956 c2 censys Havoc ROUTERHOSTING	DonPasci

Figura 4.19: Database ThreatFox.

- **OISD – Big** [38]: Versione estesa della lista OISD, integra una copertura ampia contro domini sospetti, pubblicità, tracciatori e siti potenzialmente pericolosi.

Capitolo 5

Test, analisi prestazionale e risultati del sistema integrato

5.1 Metodologia di test

La fase di testing rappresenta un momento cruciale nel ciclo di vita del software, poiché serve a validare in modo empirico le funzionalità e le prestazioni di un'architettura di sicurezza multilivello. Per garantire che il sistema integrato fosse realmente affidabile e robusto, è stato seguito un percorso di test con l'obiettivo di osservare come si comportano i vari componenti del sistema nella loro totalità, concentrandosi su due aspetti fondamentali ma tra loro collegati: da un lato le misure di sicurezza implementate localmente sul dispositivo dell'utente, e dall'altro le contromisure attive sul cloud, che proteggono l'infrastruttura di rete.

Tutti i test sono stati eseguiti utilizzando come client un sistema operativo Windows 10 con a bordo la piattaforma Java OpenJDK 23. Per le prove specifiche sulle difese di rete, questo ambiente è stato affiancato dall'uso del Windows Subsystem for Linux (WSL) con distribuzione Ubuntu 24.04. L'infrastruttura cloud, invece, si basa su un'istanza di OPNsense ospitata su Oracle Cloud Infrastructure (OCI). La verifica dei risultati è stata supportata dall'analisi dei log, dall'osservazione della UI e dall'uso di strumenti come Wireshark.

5.2 Scenari di test e risultati delle difese locali

In questo capitolo verranno descritti in maniera dettagliata i diversi scenari di test specifici per valutare le funzionalità e l'efficacia delle contromisure locali implementate nel client. Ogni scenario è stato costruito con una struttura ben definita che comprende: precondizioni, dati di input, sequenza di esecuzione, risultati attesi e, infine, un confronto con i risultati effettivamente ottenuti. È stato adottato questo approccio perché consente di garantire la ripetibilità dei test e la tracciabilità dei comportamenti osservati del sistema.

5.2.1 Rilevamento basato su firme

Il primo scenario si concentra sulla capacità del sistema di rilevare e gestire minacce note. In particolare, l'obiettivo del test è verificare che un file contenente una firma virale riconosciuta, come l'eseguibile di test `eicar.com` — un file standard non pericoloso sviluppato dall'European Institute for Computer Antivirus Research (EICAR) per simulare un virus e testare i software antivirus — venga correttamente spostato in quarantena e identificato come minaccia. Infine, il test mira a verificare che, nell'interfaccia grafica, siano visibili le azioni disponibili per l'utente, ovvero eliminare il file o ripristinarlo nella sua posizione originale.

Le condizioni iniziali prevedono che:

- il sistema sia correttamente configurato;
- il servizio antivirus dell'applicazione sia attivo e funzionante;
- ClamAV sia installato e operativo;
- il file di test `eicar.com` sia disponibile per il download.

Il test è stato articolato in due fasi principali: In un primo momento, è stato scaricato un file PDF innocuo per verificare che il motore antivirus non rilevasse minacce erroneamente (Figura 5.1).

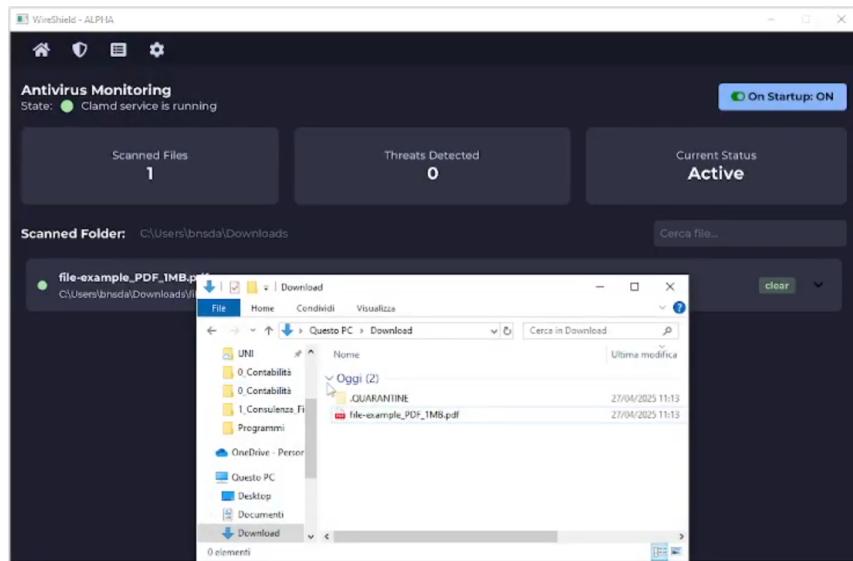


Figura 5.1: Scansione di un file PDF innocuo.

Dopo aver confermato che il file venisse riconosciuto come sicuro e non soggetto ad alcuna azione correttiva, è stata avviata la seconda fase, consistente nel download del file `eicar.com`, che invece è stato prontamente individuato come minaccia e confinato in quarantena nella directory `.QUARANTINE/` (Figura 5.2).

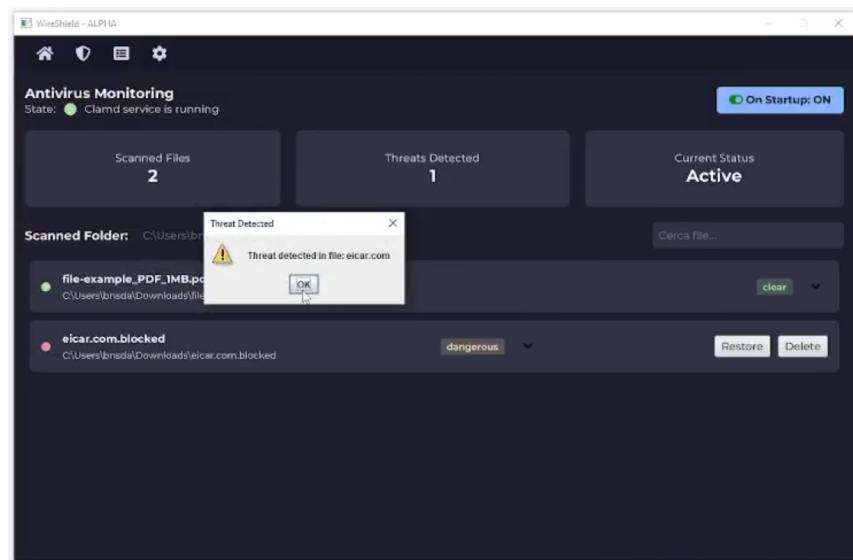


Figura 5.2: Scansione di un file EICAR.

L’interfaccia utente ha reagito notificando la presenza dell’infestazione e offrendo all’utente due opzioni: eliminare permanentemente il file o ripristinarlo nella sua posizione originale.

Il comportamento osservato ha pienamente soddisfatto le aspettative: nessun falso positivo è stato rilevato sul file PDF, mentre il file contenente la firma virale è stato gestito correttamente in ogni fase del processo. Il test può pertanto considerarsi superato.

5.2.2 Rilevamento euristico di malware modificato

Il secondo scenario di test è stato progettato per valutare la capacità del sistema di identificare varianti modificate di malware attraverso l'analisi euristica, anche in assenza di una corrispondenza esatta nel database delle firme.

L'obiettivo principale è verificare se l'applicazione sia in grado di riconoscere una versione modificata di un file EICAR noto, mettendo in atto le strategie di prevenzione previste e garantendo così una protezione anche contro minacce meno convenzionali.

Le condizioni iniziali prevedono che:

- il sistema sia correttamente configurato;
- il servizio antivirus dell'applicazione sia attivo e funzionante;
- ClamAV sia installato e operativo;
- il file di test `eicar.txt` sia disponibile per il download.

Il test è stato suddiviso in due fasi: nella prima è stato scaricato, isolato e scansionato il file originale `eicar.txt`, contenente la classica stringa di test antivirus e, come atteso, il sistema ha riconosciuto il file come noto (Figura 5.3).

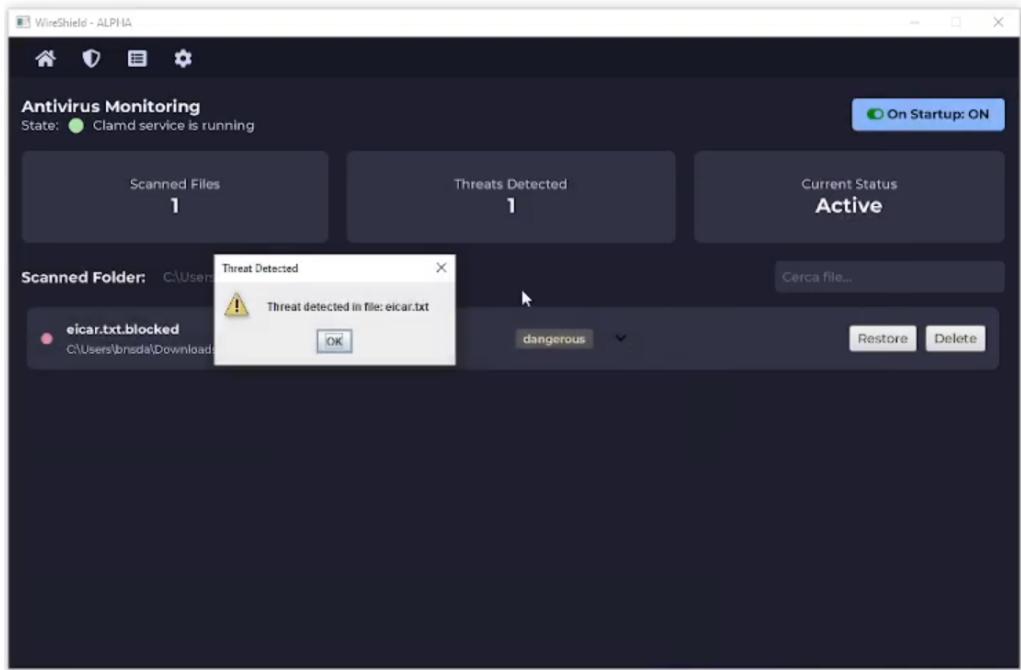


Figura 5.3: Rilevamento del file originale eicar.txt.

Nella seconda fase, il file è stato alterato con modifiche alla stringa standard EICAR (Figura 5.4), mantenendo però la struttura generale del file, al fine di testare la capacità di rilevamento euristico di ClamAV.



Figura 5.4: Contenuto modificato del file eicar_modified.txt.

Per via delle modifiche apportate, il file `eicar_modified.txt` non è stato identificato come minaccia dal motore antivirus, di conseguenza, il sistema ha ripristinato automaticamente il file nella sua posizione originale (Figura 5.5).

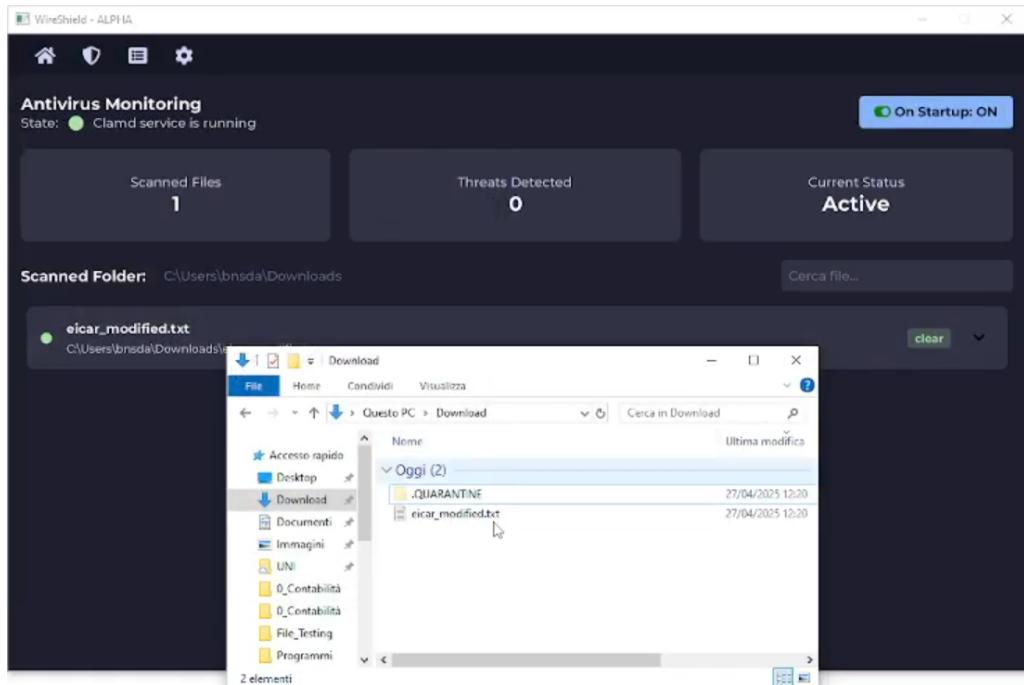


Figura 5.5: Scansione del file `eicar_modified.txt`.

Questo risultato indica che, con campioni di semplice alterazione e con le attuali configurazioni dell'applicazione, ClamAV non è riuscito a riconoscere la variante modificata del file EICAR come un potenziale malware. Di conseguenza, il test è da considerarsi fallito per quanto riguarda la capacità di rilevamento euristico nel contesto specifico.

Nota: Per migliorare i risultati di questo tipo di rilevamento, si suggerisce di impiegare campioni più complessi o di configurare ClamAV per attivare livelli di analisi euristica più approfonditi, ove disponibili.

5.2.3 Analisi statistica di falsi positivi e falsi negativi

Il terzo scenario si focalizza sulla valutazione della capacità del sistema di distinguere correttamente i file benigni dai file malevoli.

L'obiettivo è misurare con precisione il tasso di falsi positivi (file innocui segnalati erroneamente come minacce) e falsi negativi (file pericolosi non rilevati), al fine di garantirne stabilità, affidabilità e accuratezza complessiva del sistema.

Le condizioni iniziali prevedono che:

- il sistema sia correttamente configurato;
- il servizio antivirus dell'applicazione sia attivo e funzionante;
- ClamAV sia installato e operativo;
- il dataset di test, composto da file benigni e malware, sia disponibile per la scansione.

Il test prevede l'analisi di un insieme di file benigni — come `sample3.ppt`, `sample4.docx`, `sample3.csv`, `sample_640x426.gif`, `sample_640x426.bmp`, `sample3.c` — e di file malevoli contenenti la stringa virale EICAR — come `eicar.com`, `eicar.txt`, `eicar.zip`.

Durante l'esecuzione, l'applicazione ha scansionato ogni file e, successivamente, sono stati raccolti i risultati per identificare falsi positivi, falsi negativi e rilevazioni corrette.

L'analisi ha raccolto dati quali il numero totale di file scansionati, i malware rilevati, e le percentuali di errori di classificazione.

Il sistema ha riconosciuto correttamente tutti i file benigni come sicuri, mentre tutti i file contenenti la firma EICAR sono stati rilevati come minacce e sono rimasti bloccati in quarantena in attesa di una decisione da parte dell'utente (Figura 5.6, Figura 5.7).

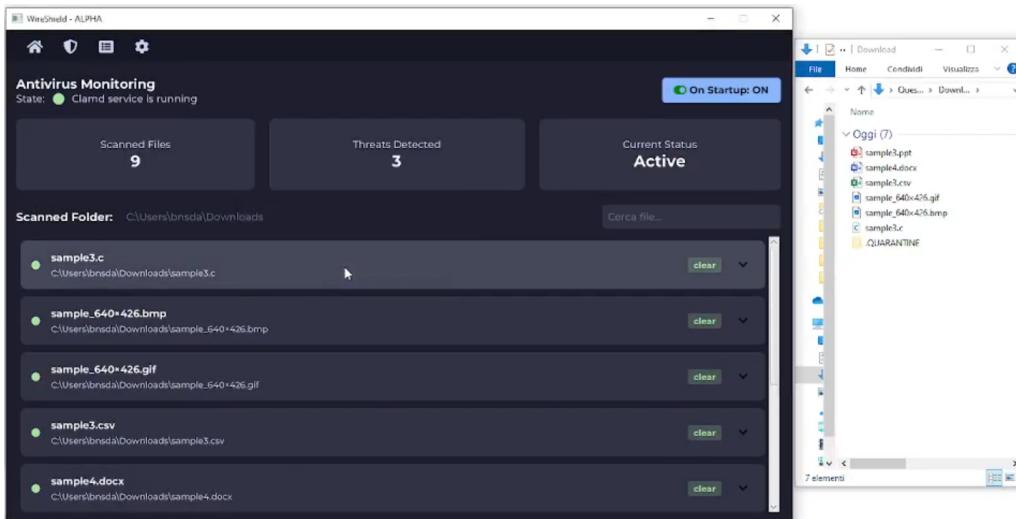


Figura 5.6: Prima parte della lista dei risultati della scansione.

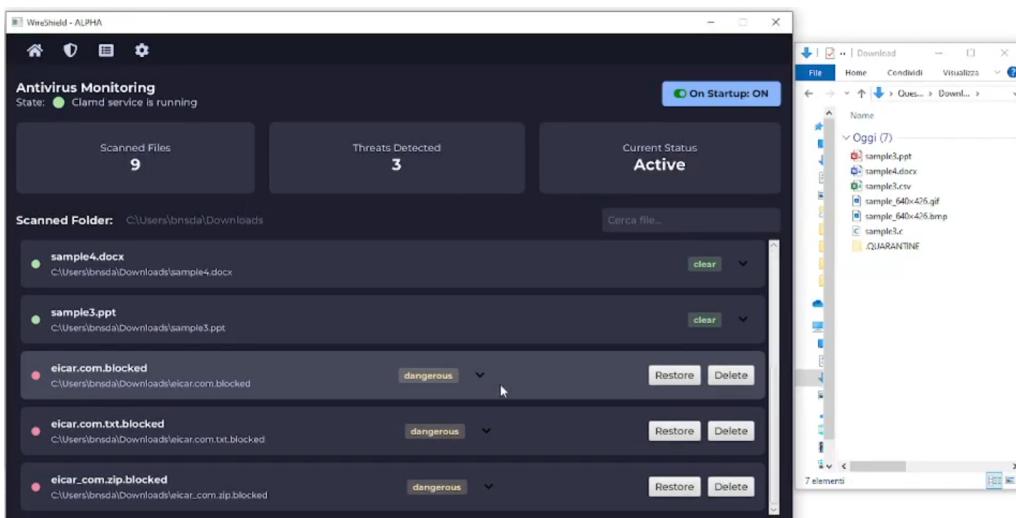


Figura 5.7: Seconda parte della lista dei risultati della scansione.

Non sono stati riscontrati falsi positivi né falsi negativi, e l'applicazione ha mantenuto la stabilità per tutta la durata del test.

Il test è pertanto considerato superato, evidenziando un buon grado di affidabilità nella distinzione tra file sicuri e malware.

Tuttavia, è importante sottolineare che la validità statistica di questo test è estremamente limitata: il numero di file analizzati è ridotto, e il dataset si basa esclusivamente su un unico tipo di malware, la stringa EICAR, ampiamente nota e facilmente rilevabile da qualsiasi motore antivirus configurato correttamente. Una valutazione

più robusta dovrebbe includere un campione significativamente più ampio e variegato, contenente malware reali, anche in forme offuscate o modificate, così da mettere realmente alla prova le capacità euristiche e comportamentali del motore antivirus ClamAV ma, per ragioni di tempo e complessità, si è scelto di non perseguire tale strada. Nonostante ciò, esistono studi più approfonditi che analizzano le potenzialità di ClamAV in contesti più realistici: tra questi, uno studio indipendente condotto da Splunk [39] nel 2022 su oltre 416.000 campioni malware ha riportato un tasso di rilevamento complessivo pari a circa 60%, con performance buone su alcuni formati come EXE, DOCX ed ELF, ma carenze evidenti su altri, tra cui XLSB, RAR e JS.

5.2.4 Performance e tempi di scansione con ClamAV

Il quarto scenario di test si concentra sulla valutazione delle prestazioni del sistema durante la scansione di file di dimensioni variabili. L'obiettivo principale è misurare i tempi necessari per completare la scansione e verificare che l'utilizzo delle risorse di sistema (CPU e memoria) rimanga entro limiti accettabili, garantendo così la stabilità e la reattività dell'applicazione anche sotto carico.

Le condizioni iniziali prevedono che:

- il sistema sia correttamente configurato;
- il servizio antivirus dell'applicazione sia attivo e funzionante;
- ClamAV sia installato e operativo;

Il test si è svolto scansionando un dataset composto da file di dimensioni differenti: un file piccolo da 10 MB, un file medio da 50 MB e due file grandi da 100 MB e 200 MB. Durante tutta la durata del test, è stato monitorato sia in tempo reale l'utilizzo di CPU e memoria, sia lo spostamento nella directory `quarantine/`, il tempo di scansione, e il relativo ripristino dei file, poiché si trattava di file innocui.

È stato verificato che il sistema non manifestasse rallentamenti, blocchi o crash, e che le operazioni di scansione, quarantena e ripristino dei file fossero gestite in modo fluido e rapido. I risultati ottenuti hanno confermato la rapida scansione e la corretta gestione dei file scansionati, con un consumo di risorse contenuto che rimane entro limiti operativi accettabili (Tabella 5.1).

Tabella 5.1: Risultati dei test di performance per la scansione di file con ClamAV.

Dimensione File	CPU (%)	Memoria Privata (KB)	Working Set (KB)	Utilizzo Fisico (%)	Tempi di Scansione (min)
10 MB	9,96	812,040	906,132	86,00	0,07
50 MB	16,84	812,040	906,132	87,89	0,07
100 MB	17,69	831,864	956,592	75,40	0,07
200 MB	14,01	813,672	956,592	80,07	0,03

I risultati possono apparire controlli intuitivi, ma questo comportamento può essere spiegato da due fattori principali:

1. L'uso del demone `clamd` consente di mantenere il database delle firme in memoria, riducendo l'overhead della scansione indipendentemente dalla dimensione del file.
2. I file testati erano appena scritti sul disco e molto probabilmente presenti nella cache del file system, consentendo a ClamAV di leggerli rapidamente dalla RAM senza attendere l'accesso al disco fisico.

Infine, differenze temporanee dovute al carico del sistema e alla gestione dei processi possono contribuire a variazioni nei tempi di scansione.

In ogni caso, i tempi di scansione rapidi e la gestione stabile di CPU e memoria confermano la robustezza e l'efficienza del sistema anche sotto carico; pertanto il test può considerarsi superato.

5.2.5 Gestione della quarantena per file infetti e puliti

L'ultimo scenario di test è finalizzato a verificare il corretto funzionamento del sistema di quarantena all'interno dell'applicazione e, in particolare, la gestione differenziata tra file infetti e file puliti. L'obiettivo è assicurarsi che i file puliti vengano ripristinati correttamente e possano essere eseguiti senza problemi. Per quanto riguarda i file riconosciuti come minacce, bisogna verificare che essi rimangano in quarantena, attendendo la decisione dell'utente di ripristinare o eliminare i file in questione.

Le condizioni iniziali prevedono che:

- il sistema sia correttamente configurato;
- il servizio antivirus dell'applicazione sia attivo e funzionante;
- ClamAV sia installato e operativo;
- il file infetto `eicar.com` e il file benigno `file-example_PDF_1MB.pdf` siano disponibili per il download.

Il test si è articolato nelle seguenti fasi: innanzitutto, è stato simulato il download del file benigno `file-example_PDF_1MB.pdf` (Figura 5.8), che, come previsto, è stato tempestivamente spostato in quarantena per l'analisi.

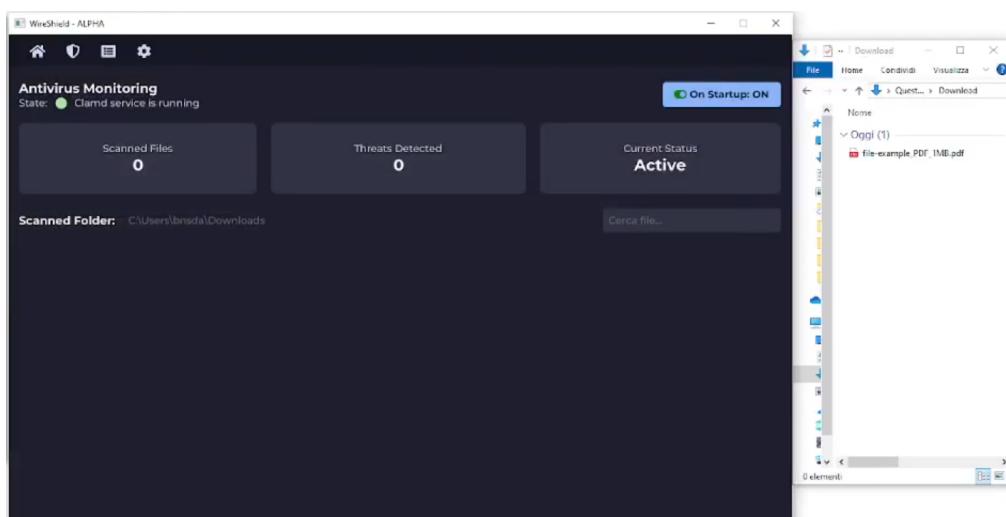


Figura 5.8: Rilevamento download del file benigno.

Durante questa fase, il file è stato rinominato aggiungendo l'estensione `.blocked` al nome originale e, al termine della scansione, poiché il file è risultato innocuo, è stato automaticamente ripristinato nel percorso originale, rimuovendo l'estensione `.blocked` e consentendone l'esecuzione completa (Figura 5.9).

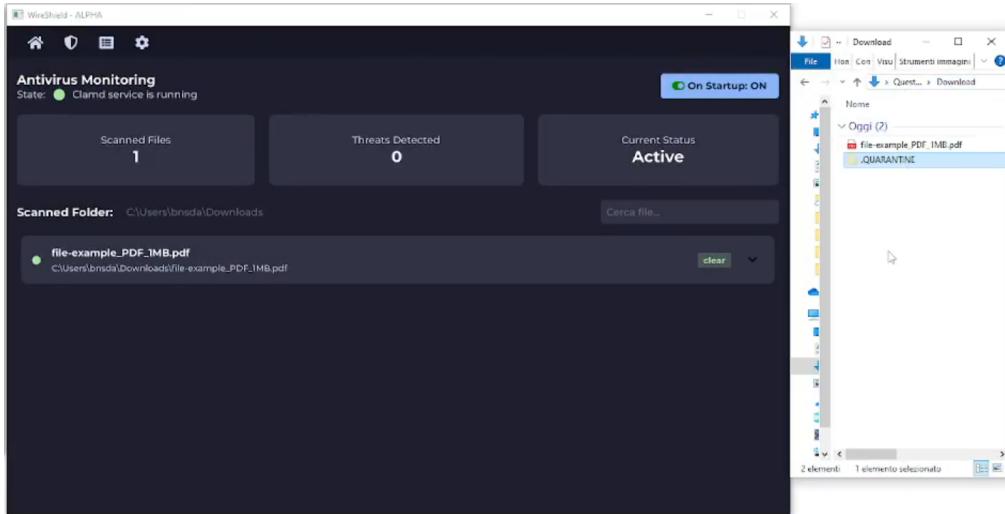


Figura 5.9: Scansione del file benigno.

Successivamente, è stato simulato il download del file infetto `eicar.com` (Figura 5.10), che è stato immediatamente spostato nella cartella di quarantena `.QUARANTINE` (Figura 5.11) e rinominato aggiungendo l'estensione `.blocked` (Figura 5.12), per poi essere rilevato come minaccia.

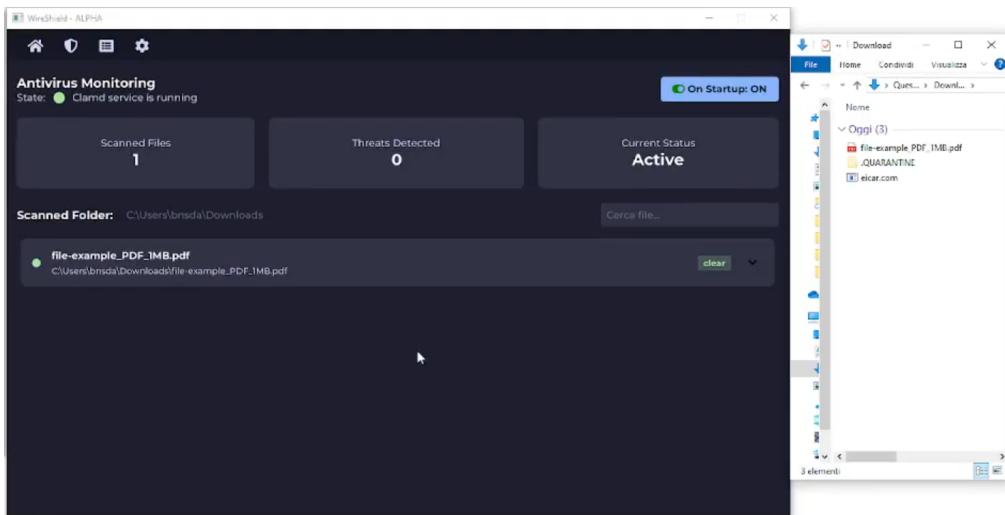


Figura 5.10: Rilevamento download del file infetto.

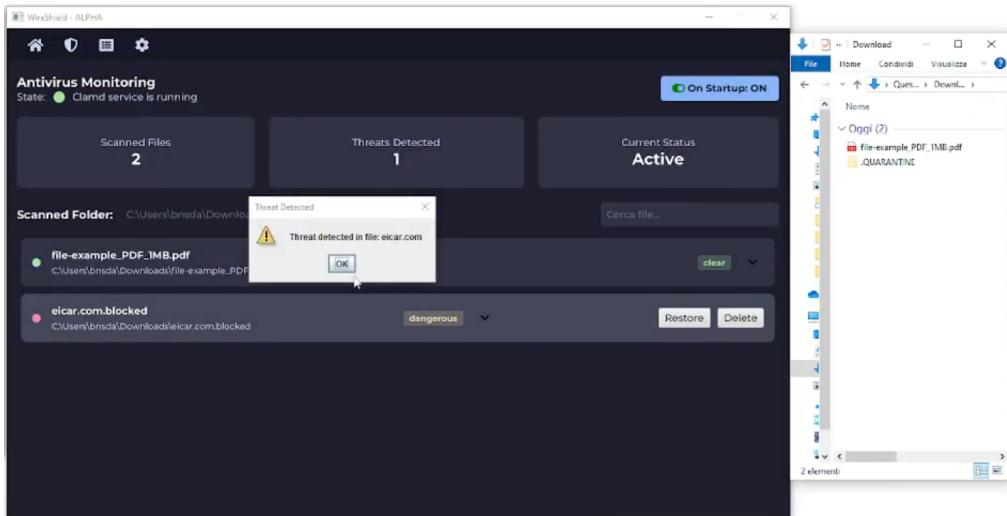


Figura 5.11: Scansione del file infetto.

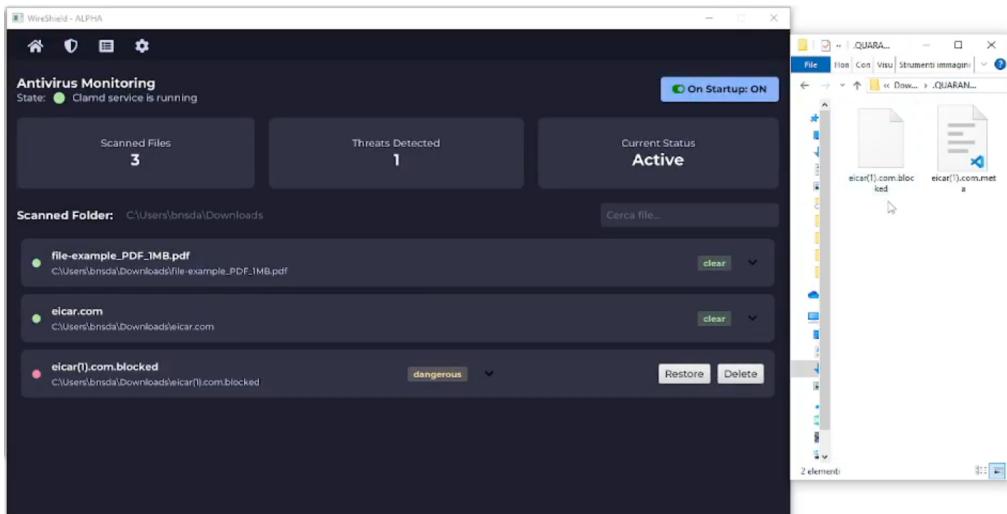


Figura 5.12: Rinominazione del file in quarantena.

Contestualmente, è stato controllato che l’interfaccia utente mostrasse all’utente due opzioni: ripristinare il file o eliminarlo definitivamente. Nel corso del test sono state eseguite due azioni principali per la verifica del funzionamento della quarantena, con i seguenti risultati:

- **Ripristino del file infetto:** tramite l’interfaccia grafica, è stato selezionato il ripristino del file messo in quarantena (Figura 5.13) e automaticamente il sistema ha reinserito il file nella sua posizione originale (Figura 5.14).

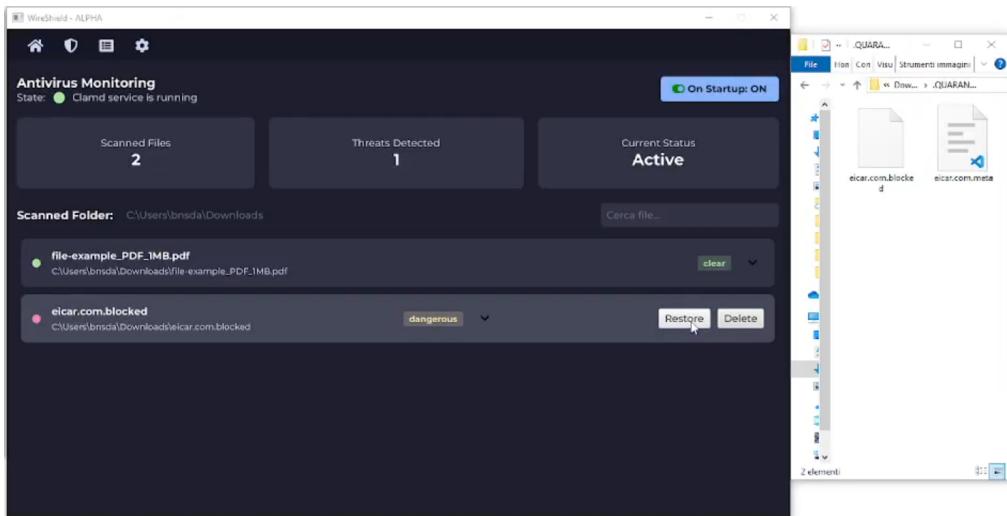


Figura 5.13: Ripristino del file infetto.

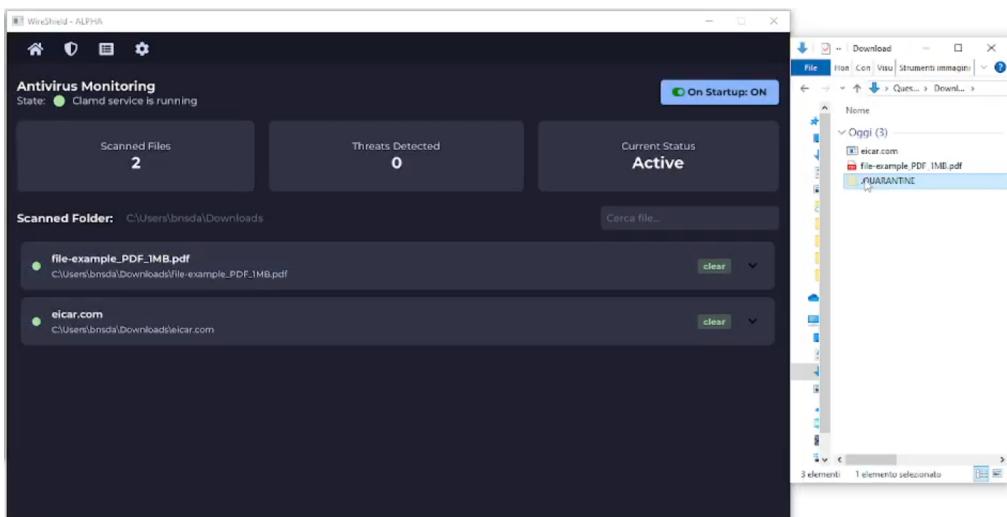


Figura 5.14: File infetto ripristinato come file pulito.

- **Nuovo tentativo di download:** successivamente, il file è stato nuovamente scaricato. Anche in questo caso, il motore antivirus lo ha rilevato come minaccia, spostandolo di nuovo in quarantena e aggiornando i metadati.
- **Eliminazione definitiva:** infine, è stata testata l'opzione di eliminazione tramite interfaccia grafica (Figura 5.15). Il file è stato rimosso dal dispositivo in modo permanente, confermando il corretto funzionamento del meccanismo di cancellazione (Figura 5.16).

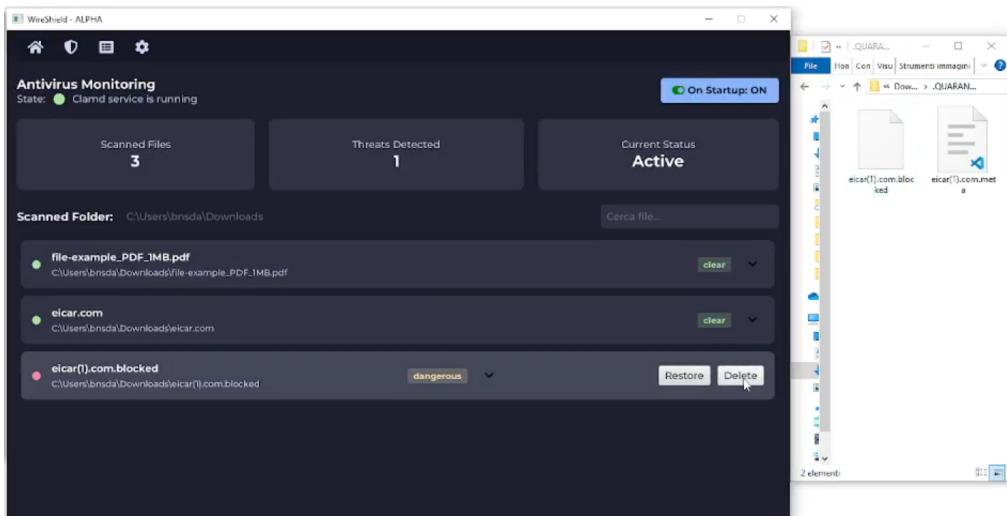


Figura 5.15: Eliminazione del file dalla quarantena.

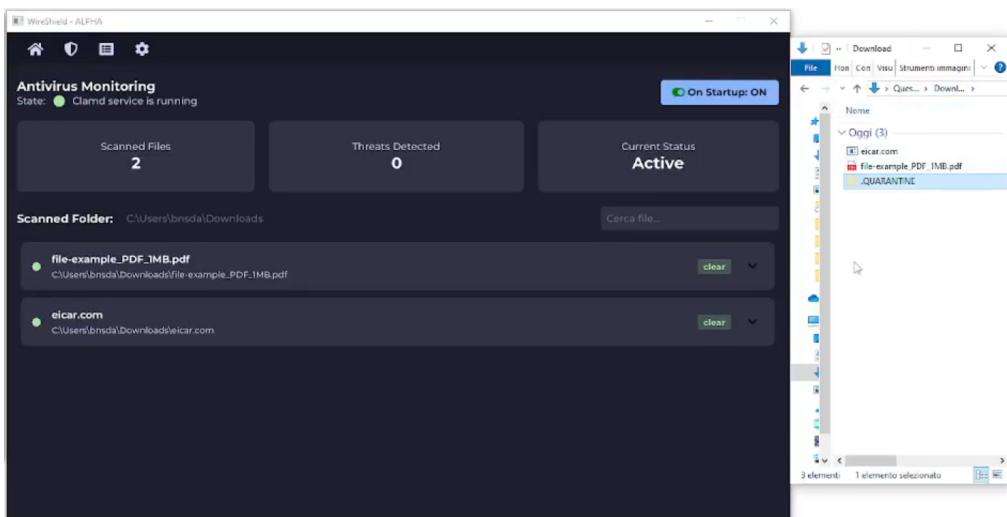


Figura 5.16: File eliminato definitivamente dalla quarantena.

Il comportamento osservato corrisponde pienamente alle aspettative: i file infetti sono rimasti bloccati in quarantena, impedendo la loro esecuzione, mentre i file puliti sono stati ripristinati correttamente, confermando l'efficacia del sistema di gestione della quarantena. Il test può pertanto considerarsi superato.

5.3 Scenari di test e risultati delle difese di rete

Così come per il client desktop, anche per il Next Generation Firewall (NGFW) è fondamentale verificare l'efficacia e il corretto funzionamento delle misure implementate.

In questo paragrafo verranno presentati cinque casi di test, ciascuno focalizzato su una specifica componente del sistema.

Per ogni scenario verranno analizzati in dettaglio i risultati ottenuti, confrontandoli con i comportamenti attesi sulla base delle specifiche teoriche, al fine di individuare eventuali discrepanze o criticità.

5.3.1 Firewall

All'interno di un sistema multilivello come quello in esame, il firewall rappresenta la prima linea di difesa, occupando il livello più esterno e basilare. La sua funzione principale è quella di eseguire un filtraggio iniziale delle connessioni in ingresso, bloccando preventivamente quelle ritenute potenzialmente pericolose sulla base di regole predefinite.

Nonostante il suo meccanismo di funzionamento sia relativamente semplice da comprendere, il firewall risulta piuttosto complesso da testare in modo rigoroso: non sarebbe efficace tentare connessioni casuali verso endpoint remoti al solo scopo di verificare il comportamento del filtro.

Per ovviare a questa difficoltà, è stato progettato un sistema di test ad hoc, basato su un'architettura client-server: un server configurato su una VPS dotata di indirizzo IPv4 pubblico espone un ampio ventaglio di porte, alcune autorizzate dalla configurazione del firewall e altre che non lo sono. Parallelamente, un client interno al sistema tenta di connettersi a ciascuna delle porte rese disponibili dal server, registrando per ogni tentativo l'esito della connessione (Figura 5.17).

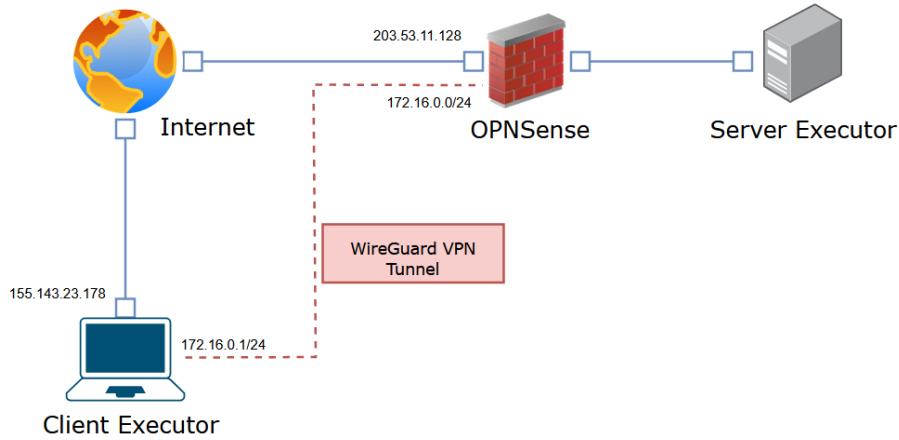


Figura 5.17: Schema di comunicazione tra client e server, nel caso di connessione VPN WireGuard attiva.

All'esecuzione, l'applicativo server produce un output, informando l'utente dei socket creati ed esposti. In particolare, il server tenta di aprire una connessione in ascolto su tutte le porte comprese tra 1 e 1024, suddividendole in due categorie: quelle autorizzate dalla configurazione del firewall (porte CONSENTITE dal NGFW) e quelle non autorizzate (porte EXTRA).

Listing 5.1: Output server

```
[*] Aggiunta di porte extra per dimostrazione (modalita : common)...
    - Aggiunte 13 porte comuni extra:
        [21, 23, 53, 69, 79, 111, 113, 135, 137, 139, 513, 514,
         515]
--- Avvio Firewall Test Server ---
[*] Porte consentite (attese):
    [22, 25, 80, 110, 143, 443, 465, 993, 995]
[*] Porte extra (dimostrative)
[*] Totale porte su cui si tentera l'ascolto: 22
[*] In ascolto sulla porta 21 (EXTRA)...
[*] In ascolto sulla porta 22 (CONSENTITA)...
[*] In ascolto sulla porta 23 (EXTRA)...
[*] In ascolto sulla porta 25 (CONSENTITA)...
[!] ERRORE: Porta 53 gia in uso.
[*] In ascolto sulla porta 69 (EXTRA)...
[*] In ascolto sulla porta 79 (EXTRA)...
[!] ERRORE: Porta 80 gia in uso.
[*] In ascolto sulla porta 110 (CONSENTITA)...
[*] In ascolto sulla porta 111 (EXTRA)...
[*] In ascolto sulla porta 113 (EXTRA)...
[*] In ascolto sulla porta 135 (EXTRA)...
```

```

[*] In ascolto sulla porta 137 (EXTRA)...
[*] In ascolto sulla porta 139 (EXTRA)...
[*] In ascolto sulla porta 143 (CONSENTITA)...
[*] In ascolto sulla porta 443 (CONSENTITA)...
[*] In ascolto sulla porta 465 (CONSENTITA)...
[*] In ascolto sulla porta 513 (EXTRA)...
[*] In ascolto sulla porta 514 (EXTRA)...
[*] In ascolto sulla porta 515 (EXTRA)...
[*] In ascolto sulla porta 993 (CONSENTITA)...
[*] In ascolto sulla porta 995 (CONSENTITA)...

-----
[*] Server avviato. Tentativo di ascolto su 22 porte.
[*] Numero attuale di listeners ATTIVI: 20
[*] Premi Ctrl+C per fermare il server.
-----
```

Al fine di verificare il corretto funzionamento del sistema di test, è stata eseguita una prima prova in assenza di connessione VPN WireGuard con il NGFW. In questa fase, l'applicativo client ha tentato di connettersi a tutte le porte esposte dal server, generando il seguente report:

Listing 5.2: Output client

```

--- Avvio Firewall Test Client ---
Inserisci l'indirizzo IP del server da testare: 152.67.76.115

[*] Inizio scansione delle porte 1-1024 su 152.67.76.115...
[*] Timeout per connessione impostato a: 0.3 secondi.
[*] Porte attese come APERTE: [22, 25, 80, 110, 143, 443, 465,
  993, 995]

-----
[*] Testando porta 80/1024...[!] Porta 80: Timeout durante la
  ricezione
  (ma connessione OK).
[*] Testando porta 111/1024...[!] Porta 111: Timeout durante
  la ricezione
  (ma connessione OK).
[*] Scansione completata.

-----
--- Report Risultati ---

[+] Porte Consentite (come atteso): 9
  [22, 25, 80, 110, 143, 443, 465, 993, 995]
[!] PORTE ATTESE APERTE MA RISULTATE CHIUSE/FILTRATE: 0
[+] Porte Bloccate Correttamente (come atteso): 1006
[!!!!] ALLARME: PORTE ATTESE BLOCCATE MA RISULTATE APERTE: 9
  [21, 23, 69, 79, 111, 113, 513, 514, 515]
```

In questo primo test il client ha individuato che tramite tutte le porte che dovevano essere bloccate dal NGFW è stato possibile connettersi al server. Si tratta di un risultato da noi atteso, in quanto la VPN WireGuard non risulta attiva, e di conseguenza il traffico non è stato instradato verso il NGFW.

Verificato il corretto funzionamento del sistema di test, è stato possibile eseguire un secondo test, assicurandoci prima di essere connessi al NGFW tramite la VPN WireGuard:

Listing 5.3: Output client

```
--- Avvio Firewall Test Client ---
Inserisci l'indirizzo IP del server da testare: 152.67.76.115

[*] Inizio scansione delle porte 1-1024 su 152.67.76.115...
[*] Timeout per connessione impostato a: 0.3 secondi.
[*] Porte attese come APERTE: [22, 25, 80, 110, 143, 443, 465,
993, 995]
-----
[*] Testando porta 80/1024...[!] Porta 80: Timeout durante la
ricezione
    (ma connessione OK).
[*] Scansione completata.
-----
--- Report Risultati ---
-----
[+] Porte Consentite (come atteso): 9
    [22, 25, 80, 110, 143, 443, 465, 993, 995]
[+] Porte Bloccate Correttamente (come atteso): 1015
```

Dall'analisi di questo risultato, emerge chiaramente che solo le connessioni dirette verso porte esplicitamente permesse hanno ricevuto risposta dal server. Nei restanti casi, l'assenza di feedback indica che le richieste non sono nemmeno arrivate a destinazione, a conferma del corretto blocco da parte del firewall (NGFW).

5.3.2 Peer isolation

La funzionalità in esame è stata introdotta per prevenire una potenziale vulnerabilità all'interno dell'infrastruttura di rete: si intende impedire che i peer appartenenti alla medesima rete virtuale WireGuard possano comunicare direttamente tra loro attraverso il Next Generation Firewall (NGFW), sfruttandolo come punto di transito. Tale scenario rappresenterebbe una grave criticità in termini di sicurezza, in quanto favorirebbe la propagazione non controllata di minacce, come malware o altre azioni malevoli, all'interno della rete stessa.

Per accertare l'effettiva efficacia di questa misura di sicurezza, è stata condotta una serie di test articolata in più fasi:

1. Verifica iniziale in assenza di connessione WireGuard

I dispositivi coinvolti sono stati impostati con l'interfaccia WireGuard disattivata, ed è stata forzata una comunicazione ICMP (*ping*) tra i due, al fine di verificare che fossero in grado di comunicare tra di loro.

Listing 5.4: Output PING

```
Esecuzione di Ping 10.0.0.114 con 32 byte di dati:  
Risposta da 10.0.0.114: byte=32 durata=77ms TTL=128  
Risposta da 10.0.0.114: byte=32 durata=131ms TTL=128  
Risposta da 10.0.0.114: byte=32 durata=217ms TTL=128  
Risposta da 10.0.0.114: byte=32 durata=131ms TTL=128  
Statistiche Ping per 10.0.0.114:  
Pacchetti: Trasmessi = 4, Ricevuti = 4,  
Persi = 0 (0% persi),  
Tempo approssimativo percorsi andata/ritorno in  
millisecondi:  
Minimo = 77ms, Massimo = 217ms, Medio = 139ms
```

2. Attivazione della connessione e abilitazione delle comunicazioni interne

Successivamente, è stata stabilita la connessione VPN WireGuard (con il NGFW) su entrambi i dispositivi e, lato NGFW, è stata temporaneamente autorizzata la comunicazione tra i peer appartenenti all'interfaccia virtuale `tunnel0` (Figura 5.18).

	Protocol	Source	Port	Destination	Port	Gateway	Schedule	Description	
<i>Automatically generated rules</i>									
▶ → ↗ ⓘ	IPv4 UDP	Tunnel0 net	*	172.16.0.1	53 (DNS)	*	*	Allow packets to local resolver	
▶ → ↗ ⓘ	IPv4+6 *	Tunnel0 net	*	Tunnel0 net	*	*	*	Block intra-peer communications	
✖ → ↗ ⓘ	IPv4+6 UDP	Tunnel0 net	*	*	53 (DNS)	*	*	Block all DNS packets outbound	
▶ → ↗ ⓘ	IPv4+6 TCP	Tunnel0 net	*	H_Cloudflare	*	*	*	guarantee TCP transit to H_Cloudflare hosts	
✖ → ↗ ⓘ	IPv4+6 TCP	Tunnel0 net	*	H_Doh	*	*	*	Block DoT services	
▶ → ↗ ⓘ	IPv4+6 TCP	Tunnel0 net	*	*	80 (HTTP)	*	*	Allow TCP HTTP	
▶ → ↗ ⓘ	IPv4+6 TCP	Tunnel0 net	*	*	443 (HTTPS)	*	*	Allow TCP HTTPS	
▶ → ↗ ⓘ	IPv4+6 UDP	Tunnel0 net	*	*	443 (HTTPS)	*	*	Allow UDP QUIC HTTPS	
▶ → ↗ ⓘ	IPv4+6 TCP	Tunnel0 net	*	*	143 (IMAP)	*	*	Allow TCP IMAP	
▶ → ↗ ⓘ	IPv4+6 TCP	Tunnel0 net	*	*	993 (IMAP/S)	*	*	Allow TCP IMAP STARTTLS	
▶ → ↗ ⓘ	IPv4+6 TCP	Tunnel0 net	*	*	110 (POP3)	*	*	Allow TCP POP3	
▶ → ↗ ⓘ	IPv4+6 TCP	Tunnel0 net	*	*	995 (POP3/S)	*	*	Allow TCP POP3 STARTTLS	
▶ → ↗ ⓘ	IPv4+6 TCP	Tunnel0 net	*	*	25 (SMTP)	*	*	Allow TCP SMTP	
▶ → ↗ ⓘ	IPv4+6 TCP	Tunnel0 net	*	*	465 (SMTP/S)	*	*	Allow TCP SMTP STARTTLS	
▶ → ↗ ⓘ	IPv4+6 TCP	Tunnel0 net	*	*	22 (SSH)	*	*	Allow TCP SSH	
▶ → ↗ ⓘ	IPv4+6 UDP	Tunnel0 net	*	*	50000 - 65535	*	*	Allow UDP Discard	
✖ → ↗ ⓘ	IPv4+6 TCP/UDP	Tunnel0 net	*	*	*	*	*	Block outbound TCP/UDP	
▶ pass									
▶ pass (disabled)									
✖ block									
✖ block (disabled)									
○ reject									
○ reject (disabled)									
								log	
								log (disabled)	
								In	
								Out	
								first match	
								last match	

Figura 5.18: Modificata regola per Tunnel0 net: deny -> allow.

In questa configurazione, è stata eseguita una comunicazione ICMP tra i dispositivi, utilizzando gli indirizzi IP assegnati all'interno della rete WireGuard, al fine di verificarne la raggiungibilità reciproca.

Listing 5.5: Output PING

```
Esecuzione di Ping 172.16.0.3 con 32 byte di dati:
Risposta da 172.16.0.3: byte=32 durata=445ms TTL=127
Risposta da 172.16.0.3: byte=32 durata=192ms TTL=127
Risposta da 172.16.0.3: byte=32 durata=62ms TTL=127
Risposta da 172.16.0.3: byte=32 durata=189ms TTL=127
Statistiche Ping per 172.16.0.3:
    Pacchetti: Trasmessi = 4, Ricevuti = 4,
                Persi = 0 (0% persi),
    Tempo approssimativo percorsi andata/ritorno in
                millisecondi:
                Minimo = 62ms, Massimo = 445ms, Medio = 222ms
```

3. Applicazione della regola di blocco

Infine, è stata riapplicata la regola firewall che inibisce le comunicazioni tra dispositivi appartenenti alla stessa rete WireGuard. Il test ICMP è stato quindi ripetuto, per verificare che le comunicazioni risultassero effettivamente bloccate.

Listing 5.6: Output PING

```
Esecuzione di Ping 172.16.0.3 con 32 byte di dati:  
Richiesta scaduta.  
Richiesta scaduta.  
Richiesta scaduta.  
Richiesta scaduta.  
Richiesta scaduta.  
Statistiche Ping per 172.16.0.3:  
Pacchetti: Trasmessi = 4, Ricevuti = 0,  
Persi = 4 (100% persi),
```

I risultati ottenuti confermano il corretto funzionamento del meccanismo di protezione: quando la regola di blocco è attiva, i dispositivi non sono in grado di comunicare tra loro; al contrario, disabilitando la regola, la comunicazione è nuovamente consentita. Questo comportamento dimostra l'efficacia della misura adottata nel prevenire l'uso improprio del firewall come ponte tra i peer interni alla rete WireGuard.

5.3.3 IPS/IDS

Come già descritto nei capitoli precedenti, il Next Generation Firewall (NGFW) utilizza *Suricata* come motore per la rilevazione e la prevenzione delle intrusioni (IDS/IPS). Questo componente riveste un ruolo fondamentale nella sicurezza del sistema, poiché analizza il traffico di rete in tempo reale alla ricerca di comportamenti sospetti o minacce note, sulla base di regole e firme costantemente aggiornate.

Al fine di testare il comportamento del sistema, si è deciso di procedere con una serie di test mirati: l'unica modalità efficace individuata consiste nell'esecuzione di tentativi controllati e casuali, con l'obiettivo di raggiungere alcune minacce conosciute, verificando il comportamento del sistema. Durante i test, sono stati generati pacchetti e richieste in grado di innescare l'attivazione delle regole di Suricata, riproducendo alcuni pattern comunemente associati a vulnerabilità note. In particolare, è stata verificata la corretta rilevazione di payload classificati nella lista URLHaus.

La procedura di test è stata articolata come segue:

1. È stato innanzitutto individuato un URL segnalato da URLHaus come attivo e associato alla distribuzione di un file malevolo (Figura 5.19).

ID:	3526919
URL:	http://195.82.147.91/key/041525-dv/directiontitle.zip
URL Status:	Online (spreading malware for 20 hours, 39 minutes)
Host:	195.82.147.91
Date added:	2025-04-26 18:47:08 UTC
Threat:	Malware download
Reporter:	DaveLikesMalware
Abuse complaint sent (?:)	Yes (2025-04-26 18:48:08 UTC to abuse[at]dedbro[dot]pro.admin[at]vaultdweller[dot]net)
Tags:	exe zip

Figura 5.19: Dettagli della IOC selezionata per il test.

2. In un primo momento, si è tentato di accedere a tale risorsa senza utilizzare la connessione VPN WireGuard, ovvero operando al di fuori del perimetro di protezione offerto dal NGFW. Questo passaggio ha confermato che l'URL risultava effettivamente raggiungibile e che il download del file veniva avviato correttamente (Figura 5.20).

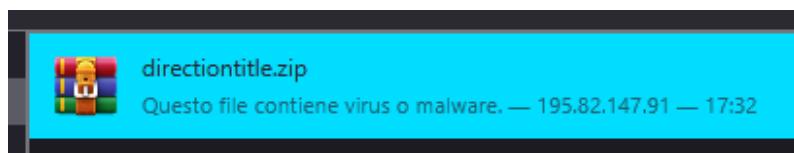


Figura 5.20: IOC scaricata con successo dal server di distribuzione.

3. Successivamente, è stata stabilita la connessione VPN WireGuard con il NGFW, ed è stato ripetuto il test di connessione.

Listing 5.7: Output cURL

```
--2025-04-27 17:40:16--  
http://195.82.147.91/key/041525-dv/directiontitle.zip  
Connecting to 195.82.147.91:80... connected.  
HTTP request sent, awaiting response...
```

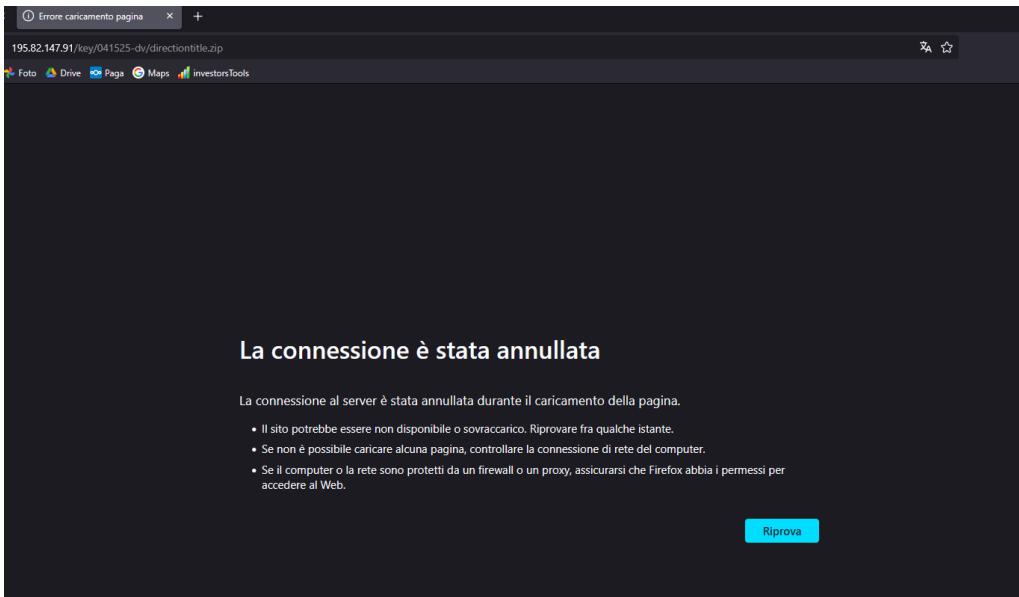


Figura 5.21: IOC non raggiungibile.

Come atteso, la connessione verso la risorsa è stata interrotta (Figura 5.21) e, osservando i log di Suricata, si può inoltre notare l'avviso che ci informa dell'individuazione della minaccia (Figura 5.22).



Figura 5.22: Avviso di individuazione della minaccia, con relativi dettagli.

Questi risultati confermano l'effettiva efficacia del motore IDS/IPS in presenza di traffico malevolo: Suricata ha correttamente generato gli alert previsti e ha provveduto al blocco del traffico secondo le regole definite.

5.3.4 DNS filtering

Il filtro DNS consiste in un resolver ricorsivo, il quale consente la risoluzione dei nomi di dominio per l'intera rete, con l'eccezione dei domini esplicitamente inclusi nelle blocklist configurate. Tali liste, aggiornate periodicamente, includono domini noti per ospitare contenuti malevoli, phishing, tracciamento o pubblicità aggressiva. Per verificare l'effettiva efficacia del filtro, sono state adottate due strategie distinte di test: una di tipo quantitativo e una qualitativa.

Test 1 – Analisi quantitativa mediante script automatizzato

La prima strategia si basa su un approccio automatizzato e quantitativo, sviluppato mediante uno script in Python in grado di effettuare richieste DNS verso un ampio insieme di domini selezionati casualmente.

I domini sono stati ottenuti tramite l'interfaccia messa a disposizione da <https://www.netmeister.org/dev/domains>, la quale estrae in modo pseudocasuale un numero definito di domini dalla *Tranco List* (<https://tranco-list.eu/>) — una classifica pubblicamente accessibile e costantemente aggiornata dei siti web più visitati a livello globale.

Listing 5.8: Output script di test

```
[*] Tentativo di recuperare 10000 domini da:  
    https://www.netmeister.org/dev/domains?n=10000  
[+] Recuperati 10000 domini.  
[*] Avvio risoluzione DNS per 10000 domini usando: 172.16.0.1  
    (max 10 workers, timeout 2.0s)...  
[*] Progresso: 1/10000 domini processati...  
[*] Progresso: 2/10000 domini processati...  
[*] Progresso: 3/10000 domini processati...  
[*] Progresso: 4/10000 domini processati...  
[*] Progresso: 5/10000 domini processati...  
[*] Progresso: 6/10000 domini processati...  
[*] Progresso: 7/10000 domini processati...  
...  
[*] Progresso: 9995/10000 domini processati...  
[*] Progresso: 9996/10000 domini processati...  
[*] Progresso: 9997/10000 domini processati...  
[*] Progresso: 9998/10000 domini processati...  
[*] Progresso: 9999/10000 domini processati...  
[*] Progresso: 10000/10000 domini processati...  
[+] Risoluzione DNS completata.
```

```

=====
===== Rapporto Dettagliato Risoluzione Domini =====
=====
Data e Ora Esecuzione: 2025-04-27 15:15:11
URL Sorgente Domini: https://www.netmeister.org/dev/domains?n=10000
Resolver DNS Utilizzato: 172.16.0.1
Timeout Query DNS: 2.0 secondi
Numero Domini Richiesti: 10000
Numero Domini Effettivamente Recuperati: 10000
Numero Domini Processati: 10000
Tempo Totale Esecuzione: 247.49 secondi
Numero Massimo Workers Utilizzati: 10
-----
----- Statistiche Risoluzione -----
-----
1. Domini Risolti (IP valido != 0.0.0.0): 8006 (80.06%)
2. Domini Non Risolti (inesist./no A/timeout/err): 1898 (18.98%)
3. Domini Bloccati (risolti a 0.0.0.0): 96 (0.96%)

```

L'esecuzione dello script ha evidenziato un comportamento coerente con le aspettative: la maggior parte dei domini risultano risolvibili correttamente, mentre quelli presenti nelle blocklist del resolver restituiscono una risposta non valida.

Test 2 – Verifica qualitativa tramite dominio noto

La seconda metodologia, adotta un'impostazione qualitativa, focalizzata sull'analisi del comportamento del sistema in presenza di un dominio noto per essere malevolo (Figura 5.23).

ID:	3527514
URL:	https://u1.pridefulamaretto.digital/gn7xlefhm.bip
URL Status:	⚠ Online (spreading malware for 8 minutes)
Host:	u1.pridefulamaretto.digital
Date added:	2025-04-27 08:39:12 UTC
Threat:	Malware download
URLhaus blocklist:	Blocked
Spamhaus DBL 🔗 :	Abused domain (malware) 🔗
SURBL 🔗 :	Blocked
Quad9 🔗 :	Blocked
AdGuard 🔗 :	Blocked
Cloudflare 🔗 :	Blocked
dns0.eu 🔗 :	Blocked
ProtonDNS 🔗 :	Blocked
Reporter:	Anonymous
Abuse complaint sent 🔗 :	Yes (2025-04-27 08:40:11 UTC to abuse@cloudflare[dot]com)
Tags:	ClearFake

Figura 5.23: Dominio scelto come test, individuato da URLHaus come malevolo.

In particolare, è stato scelto per questo test il dominio `u1.pridefulamaretto.digital`, preso casualmente dalla lista URLHaus. La procedura è stata articolata come segue:

- È stata inizialmente tentata la risoluzione del dominio senza l'ausilio della connessione VPN WireGuard, operando quindi al di fuori dalla protezione offerta dal NGFW. In questo contesto, il dominio è stato risolto correttamente.

Listing 5.9: Output dig (DNS resolution)

```
; <>> DiG 9.18.28-0ubuntu0.22.04.1-Ubuntu <>>
      u1.pridefulamaretto.digital @8.8.8.8
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id:
17714
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0,
ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;u1.pridefulamaretto.digital.           IN      A

;; ANSWER SECTION:
u1.pridefulamaretto.digital. 300 IN A
  188.114.96.7
u1.pridefulamaretto.digital. 300 IN A
  188.114.97.7

;; Query time: 40 msec
;; SERVER: 8.8.8.8#53(8.8.8.8) (UDP)
;; WHEN: Sun Apr 27 11:28:46 CEST 2025
;; MSG SIZE rcvd: 88
```

- In seguito, è stata stabilita la connessione VPN WireGuard, così da convogliare il traffico DNS attraverso il filtro gestito dal NGFW. Ripetendo la richiesta di risoluzione nello stesso contesto è stato osservato che il dominio non risultava più risolvibile in un indirizzo valido.

Listing 5.10: Output dig (DNS resolution)

```
; <>> DiG 9.18.28-0ubuntu0.22.04.1-Ubuntu <>>
      u1.pridefulamaretto.digital
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id:
56406
```

```
; ; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

; ; OPT PSEUDOSECTION:
; ; EDNS: version: 0, flags:; udp: 1232
; ; QUESTION SECTION:
; u1.pridefulamaretto.digital. IN A

; ; ANSWER SECTION:
u1.pridefulamaretto.digital. 3600 IN A 0.0.0.0

; ; Query time: 20 msec
; ; SERVER: 172.16.0.1#53(172.16.0.1) (UDP)
; ; WHEN: Sun Apr 27 10:43:50 CEST 2025
; ; MSG SIZE rcvd: 72
```

Nel complesso, i test effettuati confermano con successo la piena operatività ed efficacia del filtro DNS.

Capitolo 6

Limitazioni e possibili soluzioni

Il sistema presentato, per quanto di ampia copertura, presenta alcune limitazioni che lo rendono non adatto a tutti gli scenari di utilizzo. In questa sezione verranno discusse alcune di queste limitazioni e le possibili soluzioni per superarle.

6.1 Limitazioni del sistema

In questa sezione vengono analizzate le principali criticità del sistema, con particolare attenzione a vincoli di compatibilità, ambito di protezione, dipendenza da feed esterni e rischi legati alla centralizzazione del traffico.

- 1. Compatibilità piattaforme e motore ClamAV:** Il componente applicativo in locale è disponibile esclusivamente per ambienti Windows, rendendo il sistema non nativamente fruibile su Linux, FreeBSD, macOS e simili. Sebbene l'accesso remoto via VPN WireGuard sia teoricamente compatibile con qualsiasi sistema dotato di client WireGuard ufficiale, tale configurazione esclude però il modulo antivirus integrato, riducendo il livello di protezione complessiva. Inoltre, nella configurazione attuale, il motore ClamAV non garantisce un'analisi euristica completa, limitandosi a un controllo basato su firme statistiche. Ciò implica che le minacce emergenti o le varianti non ancora incluse nelle liste di indicatori non possano essere rilevate, riducendo l'efficacia del sistema contro malware nuovi o sconosciuti.

2. **Ambito di protezione client-side:** I feed di threat intelligence impiegati (per Suricata) sono selezionati e ottimizzati per la tutela di endpoint client; analogamente, le regole firewall sono studiate per bloccare traffico potenzialmente dannoso in uscita o in ingresso sui dispositivi utente. Ciò impedisce l'adozione del sistema come Web Application Firewall (WAF) o come protezione dedicata ai servizi server (ad esempio web server), ma trae origine dal disegno architetturale volto esclusivamente alla protezione dei client.
3. **Dipendenza dalle liste di indicatori esterni:** Gran parte delle funzionalità di rilevamento – inclusi Suricata, DNS blocklist e antivirus basati su firme – si basano su indicatori e feed di threat intelligence forniti da terze parti. La quantità, la qualità e l'aggiornamento di tali sorgenti determinano direttamente l'efficacia nel riconoscimento di minacce note, limitando la capacità del sistema di intercettare malware emergenti o varianti non ancora incluse nelle liste.
4. **Centralizzazione del traffico e single point of failure:** L'architettura VPN obbliga a instradare tutto il traffico dei client verso un'unica istanza remota, configurata come gateway. In caso di indisponibilità, guasti o manutenzioni di questo nodo, i dispositivi non possono più accedere né al sistema di sicurezza né a internet; per ripristinare la connettività, l'utente è costretto a disattivare la protezione o spegnere temporaneamente il client, esponendo il dispositivo a potenziali rischi.

6.2 Possibili soluzioni

Di seguito vengono proposte le contromisure e le strategie di mitigazione corrispondenti alle criticità individuate, volte a estendere la copertura funzionale, migliorare la resilienza e garantire un livello di difesa più robusto e flessibile.

1. Compatibilità piattaforme:

- *Porting multipiattaforma*: riscrivere e ricompilare il client locale, sostituendo eventuali dipendenze Windows-specifiche con librerie cross-platform, così da supportare nativamente Linux, macOS e FreeBSD.
- *Modularizzazione antivirus*: estrarre il motore antivirus in un servizio remoto o micro-servizio indipendente dal sistema operativo, accessibile via API, così da erogare la scansione anche ai client non-Windows.
- *Analisi euristica avanzata*: modificare l'attuale configurazione o integrare un motore di analisi comportamentale o euristica che operi in parallelo al motore ClamAV, per rilevare minacce sconosciute basate su pattern comportamentali piuttosto che su firme statiche.

2. Ambito di protezione client-side:

- *Estensione delle regole Suricata*: integrare feed e regole specifiche per la protezione dei server all'interno della stessa istanza Suricata o in istanze separate dedicate.
- *Profili configurabili*: introdurre profili “client” vs. “server” che abilitino automaticamente gruppi di regole e policy firewall in base al ruolo del nodo.

3. Dipendenza dalle liste di indicatori esterni:

- *Threat hunting e anomaly detection*: implementare un motore di behavioral analytics, basato su machine learning o regole euristiche, per rilevare attività sospette anche senza firme note.
- *Feed multi-vendor e open source*: aggregare feed commerciali e comunitari (MISP, Abuse.ch, AlienVault OTX, ecc.) con rinnovo automatico e validazione incrociata per migliorarne la copertura e la ridondanza.
- *Threat intelligence interna*: utilizzare un SIEM per raccogliere e condividere indicatori emergenti generati dalla rete interna, alimentando dinamicamente le liste di blocco.

4. Centralizzazione del traffico e single point of failure

- *Architettura geo-ridondata*: distribuire più gateway VPN in differenti regioni/data-center, configurando il client con una lista di endpoint di fallback.
- *Split-tunnel selettivo*: in caso di guasto del gateway primario, consentire lo split-tunneling per traffico non critico (es. navigazione web), mantenendo in VPN solo applicazioni sensibili.
- *Health-check e fail-over automatico*: integrare un servizio di monitoraggio che rilevi la raggiungibilità dei gateway e riconfiguri dinamicamente il client VPN (via API o DNS SRV) per inoltrare il traffico verso i nodi attivi.

Capitolo 7

Conclusioni

In questo documento ci siamo focalizzati sulla progettazione e implementazione di un sistema di sicurezza per la protezione di endpoint client, concentrandoci sulle decisioni tecniche, articolando le scelte fatte e i test finali che ne attestano il funzionamento. L'architettura proposta si basa su un client leggero ma completo, eseguibile in locale sui dispositivi utente, che permette la connessione ad un gateway remoto tramite VPN (WireGuard). Il gateway funge da punto centrale per l'analisi del traffico di rete, integrando un sistema IPS/IDS(Suricata), regole firewall dedicate e un resolver DNS con funzionalità di blocco basato su blocklist (Unbound). Inoltre, il client incorpora un modulo antivirus (ClamAV) per la scansione dei file in tempo reale e funzionalità di splitting del traffico, offrendo una protezione multi-livello contro minacce note e potenzialmente nuove.

Per comprovare l'efficacia del sistema, sono stati condotti test mirati che hanno simulato scenari di attacco comuni, come il download di file infetti, l'accesso a siti web malevoli e la trasmissione di traffico tra i dispositivi della rete. Questi test hanno permesso di valutare la capacità del sistema di rilevare e bloccare minacce in tempo reale, nonché di analizzare le prestazioni complessive e l'impatto sulla user experience. Essi hanno coperto tutti gli scenari di utilizzo, focalizzandosi sull'effettiva entrata in funzione dei meccanismi di difesa, in modo coerente con le policy impostate e con le configurazioni previste, dimostrando una complessiva prontezza nella risposta alle minacce individuate.

I risultati si presentano coerenti con le aspettative, anche se la mancanza di campioni con elevato numero di elementi, principalmente lato locale, non permette di trarre conclusioni definitive sull'effettiva efficacia del sistema. Infatti, l'accesso a virus complessi e maggiori statistiche di utilizzo effettivo avrebbero permesso di testare in modo più approfondito le capacità del sistema, evidenziando eventuali punti deboli o aree di miglioramento. Ci riteniamo tuttavia soddisfatti dei risultati ottenuti, i quali evidenziano l'effettiva efficacia del sistema anche utilizzando threat intelligence open-source. Questo evidenzia un ampio margine di miglioramento, ottenibile attraverso l'incremento della qualità dei dati attualmente utilizzati nelle liste di firme malware (ClamAV), nei pattern malevoli (Suricata) e nelle DNS blocklist (Unbound). Nonostante i risultati positivi, il sistema presenta alcune limitazioni che ne condizionano l'adozione in scenari più ampi o complessi. Tra queste, la dipendenza da feed di threat intelligence esterni, la compatibilità limitata a piattaforme Windows per il client locale, l'ambito di protezione focalizzato sui soli endpoint client e la centralizzazione del traffico che introduce un potenziale single point of failure. Queste criticità sono state analizzate in dettaglio e sono state proposte possibili soluzioni e strategie di mitigazione per superarle.

Bibliografia

- [1] Clusit. *Rapporto Clusit 2025 sulla Sicurezza ICT in Italia e nel mondo*. Rapp. tecn. 2025. URL: <https://www.Clusit.it/rapporto-Clusit/>.
- [2] Gary Stoneburner, Alice Goguen e Alexis Feringa. *Guide for Conducting Risk Assessments*. Rapp. tecn. Special Publication 800-30 Revision 1. NIST, 2012. URL: <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-30r1.pdf>.
- [3] European Union Agency for Cybersecurity. *ENISA Threat Landscape 2023*. Rapp. tecn. 2024. URL: <https://www.enisa.europa.eu/publications/enisa-threat-landscape-2023>.
- [4] Council on Foreign Relations. *Cyber Operations Tracker*. 2024. URL: <https://www.cfr.org/cyber-operations/>.
- [5] S. Kumar et al. «A Review of Cyber Threat Actors: Taxonomy, Techniques and Motivations». In: *Applied Sciences* 10.12 (2020), p. 4334. URL: <https://www.mdpi.com/2076-3417/10/12/4334>.
- [6] Canadian Centre for Cyber Security. *National Cyber Threat Assessment 2025–2026*. Rapp. tecn. 2025. URL: <https://www.cyber.gc.ca/en/guidance/national-cyber-threat-assessment-2025-2026>.
- [7] Flashpoint. *Cybercrime-as-a-Service (CaaS): The New Threat Landscape*. 2023. URL: <https://flashpoint.io/intelligence-101/threat-actor/>.
- [8] Audrey Alexander e Bennett Clifford. «Doxing and Defacements: Examining the Islamic State's Hacking Capabilities». In: (2019). URL: <https://ctc>.

- westpoint.edu/doxing-defacements-examining-islamic-states-hacking-capabilities/.
- [9] Eugene H Spafford. «The internet worm program: an analysis». In: *ACM SIGCOMM Computer Communication Review* 19.1 (1989), pp. 17–57. URL: <https://dl.acm.org/doi/10.1145/66093.66095>.
 - [10] Trend Micro. *VBS_LOVELETTER - Threat Encyclopedia*. 2000. URL: https://www.trendmicro.com/vinfo/us/threat-encyclopedia/archive/malware/vbs_loveletter.
 - [11] Matt Bishop. «Analysis of the ILOVEYOU Worm». In: *University of California, Davis* (2000). URL: https://www.researchgate.net/publication/240749154_Analysis_of_the_ILOVEYOU_Worm.
 - [12] Symantec Security Response. *W32.Stuxnet Dossier*. Rapp. tecn. Symantec Corporation (now Broadcom), 2011. URL: <https://docs.broadcom.com/docs/security-response-w32-stuxnet-dossier-11-en>.
 - [13] Council on Foreign Relations. *Connect the Dots on State-Sponsored Cyber Incidents - Operation Aurora*. 2010. URL: <https://www.cfr.org/cyber-operations/operation-aurora>.
 - [14] SolarWinds: State-sponsored global software supply chain attack. Danish Centre for Cyber Security (CFCS), 2021. URL: <https://www.cfcs.dk/globalassets/cfcs/dokumenter/rapporter/en/CFCS-solarwinds-report-EN.pdf>.
 - [15] Yeshiva University Monroe College. *The Evolution of Cyber Threats*. 2023. URL: <https://online.yu.edu/katz/blog/the-evolution-of-cyber-threats>.
 - [16] Cloudflare. *DDoS Threat Report Q4 2023*. Rapp. tecn. 2023. URL: <https://blog.cloudflare.com/it-it/ddos-threat-report-2023-q4/>.
 - [17] Cloudflare. *Cloudflare DDoS threat report for 2025 Q1*. Rapp. tecn. 2025. URL: <https://blog.cloudflare.com/ddos-threat-report-for-2025-q1/>.

- [18] Verizon. *2022 Data Breach Investigations Report*. Rapp. tecn. 2022. URL: <https://www.verizon.com/business/resources/reports/2022-dbir-data-breach-investigations-report.pdf>.
- [19] CyberInt. *Ransomware Annual Report 2024*. Rapp. tecn. 2024. URL: <https://cyberint.com/blog/research/ransomware-annual-report-2024/>.
- [20] Sophos. *The State of Ransomware 2025*. Rapp. tecn. 2025. URL: <https://assets.sophos.com/X24WTUEQ/at/9brgj5n44hqvgsp5f5bqcps/sophos-state-of-ransomware-2025.pdf>.
- [21] National Institute of Standards e Technology. *Cybersecurity Glossary*. 2023. URL: <https://csrc.nist.gov/glossary/term/cybersecurity>.
- [22] Fortinet. *Intrusion detection system*. URL: <https://www.fortinet.com/it/resources/cyberglossary/intrusion-detection-system>.
- [23] Cisco. *What Is SIEM?* URL: <https://www.cisco.com/site/us/en/learn/topics/security/what-is-siem.html>.
- [24] Fortinet. *Definizione di IPS*. URL: <https://www.fortinet.com/it/resources/cyberglossary/what-is-an-ips>.
- [25] Cisco. *What is a Firewall?* URL: <https://www.cisco.com/site/us/en/learn/topics/security/what-is-a-firewall.html>.
- [26] Fortinet. *What is SOAR?* URL: <https://www.fortinet.com/resources/cyberglossary/what-is-soar>.
- [27] Fortinet. *What is EDR?* URL: <https://www.fortinet.com/resources/cyberglossary/what-is-edr>.
- [28] Cloudflare. *what-is-dns-filtering*. URL: <https://www.cloudflare.com/it-it/learning/access-management/what-is-dns-filtering/>.
- [29] Cloudflare. *What is HTTPS Inspection?* URL: <https://www.cloudflare.com/it-it/learning/security/what-is-https-inspection/>.
- [30] Oracle Cloud Infrastructure. *Best practices for hybrid and multicloud OCI networking design*. URL: <https://docs.oracle.com/en/solutions/oci-best-practices-networking/define-workload-requirements1.html>.

- [31] Oracle Cloud Infrastructure. *Virtual Cloud Network (VCN)*. URL: <https://www.oracle.com/it/cloud/networking/virtual-cloud-network/>.
- [32] Maurice Walker. *OPNsense aarch64 firmware repository*. URL: <https://opnsense-update.walker.earth/>.
- [33] *WireGuard*. URL: <https://www.wireguard.com/>.
- [34] URL: <https://suricata.io/>.
- [35] proofpoint. *TECH BRIEFET Category Descriptions*. URL: <https://tools.emergingthreats.net/docs/ETPro%20Rule%20Categories.pdf>.
- [36] URL: <https://github.com/NLnetLabs/unbound>.
- [37] Fabio Martignon. *Application Level*. 2023. URL: <https://cs.unibg.it/martignon/documenti/rim/Application-Level.pdf>.
- [38] URL: <https://oisd.nl/includeLists/big/0>.
- [39] Ryan Kovar. *How Good is ClamAV at Detecting Commodity Malware?* 2022. URL: https://www.splunk.com/en_us/blog/security/how-good-is-clamav-at-detecting-commodity-malware.html.
- [40] *WireGuard repo*. URL: <https://github.com/wireguard>.