

MODULO 11

Malware analysis con AI

Classificazione statica e dinamica, sandbox e reverse engineering assistito dall'AI, dall'analisi del campione alle azioni di contenimento.

Ghidra

CAPEv2 / ANY.RUN

REMnux / FLARE VM

Abstract e obiettivi formativi

Abstract

Il modulo tratta l'analisi del software malevolo con il supporto dell'AI: formati PE/ELF e offuscamento, feature per la classificazione ML e relativi limiti, sandbox e tecniche di evasione, reverse engineering assistito da LLM, estrazione automatica di configurazioni e IoC, in funzione della risposta all'incidente.

Obiettivi formativi (Bloom)

- **Applicare:** condurre un'analisi statica e dinamica di base in ambiente isolato
- **Applicare:** impiegare funzioni AI di decompilazione e annotazione valutandone l'affidabilità
- **Analizzare:** interpretare i verdetti ML dei motori di classificazione

Traccia core

01 02 07

Detection & Response

03 04 05 06

o

Digital Forensics

08 09 10 11

o

Sicurezza dell'AI

06 12 13 14

Traccia core

15 16

Letture di raccordo: § 3.9, 4.8, 6.6

L'esame di un campione sospetto



Feature statiche e loro fragilità

Cosa osserva un classificatore

- Struttura PE/ELF: sezioni, import, entropia
- Stringhe, API richiamate, firme di packer
- Byte n-gram e rappresentazioni apprese del binario
- Verdetti in millisecondi su grandi volumi di campioni, alla base dei motori ML degli EDR

Cause di errore

- Packing e offuscamento alterano le feature osservate
- Concept drift: le famiglie evolvono più in fretta del training set
- Adversarial sample costruiti per evadere il classificatore (Modulo 13)
- Il verdetto ML va confermato con l'analisi dinamica

Sandbox e tecniche di evasione

Cosa osserva la sandbox

- Chiamate di sistema, processi figli, persistenza
- Traffico di rete e tentativi di C2
- Modifiche a file system e registro
- CAPEv2 e ANY.RUN producono report comportamentali; il machine learning raggruppa i comportamenti in famiglie

Tecniche di evasione

- Rilevamento della VM: artefatti hardware, driver, tempi
- Ritardi d'esecuzione e trigger condizionali su data, geografia o interazione
- Payload attivato solo fuori dall'ambiente di analisi
- Contromisure: ambienti realistici, esecuzioni prolungate, esecuzioni ripetute

L'LLM accanto al decompilatore

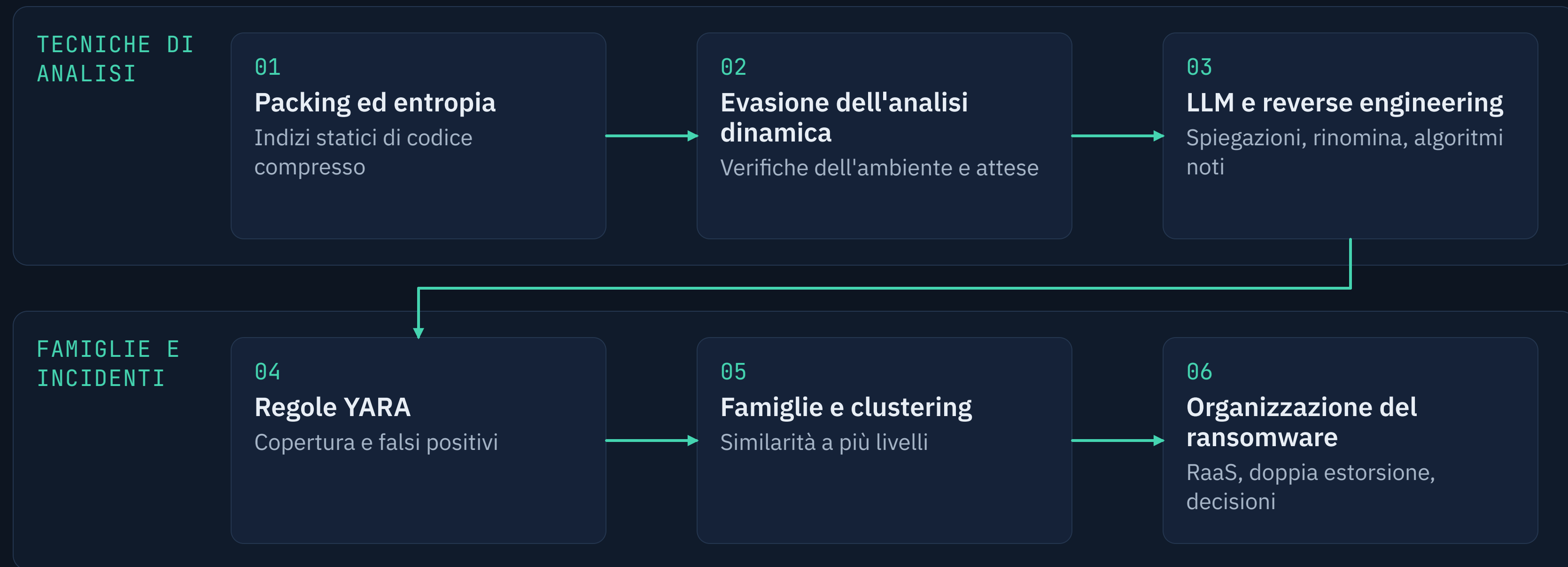
Contributi

- Spiegazione in linguaggio naturale di funzioni decompilate (Ghidra con estensioni basate su LLM)
- Rinomina simbolica coerente di variabili e funzioni
- Sintesi del comportamento complessivo del campione
- Estrazione di configurazioni e IoC per la detection a valle

Errori tipici

- Spiegazioni plausibili ma errate su codice offuscato
- Invenzione di scopi per funzioni ambigue
- Crittografia custom descritta come standard nota
- Contromisura: verificare ogni affermazione sul disassemblato; il modello non è mai citato come fonte nel report

Analisi tecnica e quadro dell'incidente



Packing ed entropia nell'analisi statica

Un **packer** comprime o cifra il codice originale e lo racchiude in uno stub che lo ricostruisce in memoria all'avvio: l'analisi statica osserva quindi l'involucro e non il programma. L'indicatore più semplice è l'**entropia** di Shannon delle sezioni: dati compressi o cifrati hanno valori vicini a 8 bit per byte, mentre codice macchina e testo si collocano più in basso. Altri indizi concordanti sono una tabella degli import ridotta a poche funzioni, spesso LoadLibrary e GetProcAddress, e nomi di sezione insoliti. Il codice effettivo diventa osservabile durante l'esecuzione oppure estraendolo dalla memoria del processo (Modulo 9).

Scenario didattico

Si consideri un campione con una sezione di codice molto ridotta e una sezione dati di 2 MB con entropia 7,9, un profilo tipico di loader. Dopo l'esecuzione in ambiente controllato, il dump della memoria del processo mostra il payload decompresso, con centinaia di stringhe leggibili, indirizzi di wallet e l'URL del server di comando e controllo, assenti nel file su disco.

Ruolo dell'AI

I classificatori basati su n-gram di byte, struttura PE e import eseguono il triage di grandi volumi di campioni; sui file compressi apprendono soprattutto le caratteristiche del packer e si degradano con packer assenti dai dati di addestramento. Sono inoltre il bersaglio degli attacchi di evasion trattati nel Modulo 13.

Sandbox ed evasione dell'analisi dinamica

Una sandbox come CAPEv2 esegue il campione in una macchina strumentata e registra chiamate di sistema, file creati, chiavi di registro e traffico di rete. Molti campioni verificano l'ambiente prima di agire. Cercano indizi di virtualizzazione, come driver, indirizzi MAC e dimensione del disco, misurano i tempi di esecuzione e controllano segni di attività umana, come movimenti del mouse, documenti recenti e tempo di attività del sistema. In alternativa attendono più a lungo della finestra di osservazione o una condizione esterna, come una data o la lingua del sistema. Un report privo di comportamenti malevoli può quindi indicare un file innocuo oppure un campione che ha riconosciuto l'ambiente di analisi.

Scenario didattico

Un dropper non mostra alcuna attività in tre esecuzioni in sandbox. Alla quarta, con orologio virtuale accelerato e un profilo utente realistico che comprende documenti, cronologia del browser e quaranta minuti di attività, scarica il payload. La differenza di comportamento a parità di campione indica una probabile evasione, da confermare individuando nel codice le verifiche sull'ambiente.

Metodo

Prima di un giudizio di innocuità si variano ambiente, durata e stimoli in più esecuzioni e si confrontano i risultati con l'analisi statica e con quella della memoria. I modelli che riassumono i report di sandbox ereditano il problema, perché sintetizzano solo i comportamenti che il campione ha mostrato.

Modelli linguistici nel reverse engineering

Il decompilatore di Ghidra produce pseudocodice C in genere fedele alla logica del programma, ma senza garanzia di correttezza e difficile da leggere: funzioni con nomi automatici formati da FUN_ e dall'indirizzo, variabili anonime, costrutti tipici del compilatore. Su questo livello un modello linguistico può spiegare in linguaggio naturale che cosa fa una funzione, proporre nomi significativi per funzioni e variabili, riconoscere algoritmi noti come RC4, Base64 o implementazioni di AES e riassumere blocchi di logica. Il risparmio di tempo riguarda il lavoro meccanico; la ricostruzione dell'architettura del campione e la valutazione delle scelte dell'autore restano compiti dell'analista.

Scenario didattico

Una funzione di 120 righe contiene un ciclo con operazioni XOR e una tabella di 256 byte. Il modello la riconosce come RC4 e suggerisce che il buffer passato in ingresso sia la chiave. L'analista verifica l'ipotesi, estrae la chiave, decifra la configurazione del campione e ricava i domini di comando e controllo.

Limiti

Il modello può riconoscere schemi inesistenti, per esempio attribuire a una cifratura personalizzata il nome di un algoritmo standard, oppure inventare uno scopo plausibile per una funzione ambigua. Ogni ipotesi va verificata sul disassemblato o con l'esecuzione. Inviare codice del campione a un servizio esterno significa inoltre far uscire dati d'indagine dal perimetro (Modulo 6, approfondimento 3).

Regole YARA per descrivere una famiglia

YARA è il linguaggio con cui gli analisti descrivono le caratteristiche che identificano un malware: stringhe, sequenze di byte con caratteri jolly per le parti variabili, proprietà strutturali del file, combinate in condizioni logiche. Per generalizzare, una regola descrive ciò che l'autore difficilmente modifica tra una versione e l'altra, come una routine distintiva o il formato della configurazione, e lascia da parte dettagli contingenti come hash o URL, che cambiano a ogni ricompilazione. La regola va verificata in due direzioni: sui campioni noti della famiglia, per misurarne la copertura, e su un ampio insieme di file legittimi, per misurarne i falsi positivi, che in produzione provocano quarantene e interruzioni di servizio.

Scenario didattico

L'analisi dell'approfondimento 3 ha messo in luce una routine RC4 con una tabella di inizializzazione non standard. La regola codifica 24 byte di quella routine con due caratteri jolly sulle parti che variano tra le build. Applicata all'archivio interno dei campioni, individua 17 file della stessa famiglia raccolti in due anni, tre dei quali non erano mai stati collegati tra loro.

Ruolo dell'AI

Un modello linguistico produce bozze di regole a partire da un campione o da un report, ma tende a scegliere stringhe fragili o troppo comuni. La scelta degli elementi distintivi richiede la conoscenza della famiglia: le proposte del modello vengono selezionate dall'analista e la regola viene provata sui due insiemi di file prima del rilascio.

Famiglie di malware e clustering

I campioni derivano spesso gli uni dagli altri e condividono librerie, generatori (builder) e codice sorgente diffuso pubblicamente. Il **clustering** li raggruppa in famiglie con misure di similarità a più livelli: fuzzy hashing sul file (Modulo 8), similarità tra funzioni del codice disassemblato, sovrapposizione dei comportamenti osservati in sandbox, riuso dell'infrastruttura di rete. Riconosciuta la famiglia, si riutilizzano analisi già svolte, come capacità note, formato della configurazione e strumenti di estrazione, e i tempi di conferma si riducono. La famiglia è un dato tecnico misurabile; l'identità di chi la usa è un'inferenza, da formulare con il grado di confidenza descritto nel Modulo 4.

Scenario didattico

Il loader di un incidente ha valori di ssdeep bassi rispetto all'archivio. Sul codice estratto dalla memoria, il confronto tra funzioni rileva circa il 70% di codice in comune con una famiglia nota: il campione era stato ricompilato e compresso con un altro packer, per cui il fuzzy hash sul file non mostrava somiglianze. L'estrattore di configurazione della famiglia restituisce i server di comando e controllo.

Cautele

Il riuso del codice può trarre in inganno. La diffusione del codice sorgente del ransomware Babuk nel 2021 ha dato origine a varianti usate da gruppi privi di legami con gli autori originali; anche le librerie comuni aumentano la similarità apparente.

Organizzazione economica del ransomware

Il ransomware attuale è organizzato con una divisione del lavoro: sviluppatori che offrono il malware in abbonamento (**ransomware as a service**, RaaS), intermediari che vendono accessi iniziali alle reti (initial access broker), affiliati che conducono l'intrusione, negoziatori e siti di pubblicazione dei dati sottratti. Il modello prevalente è la **doppia estorsione**: i dati vengono esfiltrati prima della cifratura, così che il ripristino da backup non elimini la minaccia di pubblicazione. Per l'analista le domande rilevanti riguardano quali dati sono stati sottratti, da quando l'attaccante era presente e con quali tecniche ha operato; le risposte provengono soprattutto dall'analisi di memoria, timeline e rete (Modulo 9) e orientano le decisioni legali e di notifica (Modulo 15).

Scenario didattico

La nota di riscatto identifica la famiglia, e i log di rete mostrano 60 GB trasferiti verso un servizio di archiviazione cinque giorni prima della cifratura. La timeline colloca l'accesso iniziale tre settimane prima, tramite una VPN priva di autenticazione a più fattori. Il quadro per i decisori comprende dati esfiltrati (artt. 33–34 GDPR), durata della permanenza e vettore da chiudere.

Metodo

In alcuni casi i file si recuperano senza pagare: implementazioni crittografiche difettose, chiavi ancora presenti in memoria (Modulo 7, approfondimento 3), strumenti di decifratura pubblicati dal progetto No More Ransom. La verifica di queste possibilità è un compito di analisi; l'eventuale pagamento resta una decisione dei vertici, con valutazioni legali ed etiche.

Estrazione di IoC da un campione didattico disarmato

Svolgimento in REMnux e FLARE VM

1. Analisi statica con Ghidra e assistente LLM locale: struttura, stringhe, funzioni principali
2. Detonazione in CAPEv2 e lettura del report comportamentale
3. Confronto tra previsioni dell'analisi statica, comportamenti osservati e affermazioni dell'assistente, con gli errori del modello
4. Estrazione degli IoC e proposta di regola di detection

Consegna

- Mini-report del campione: comportamento e famiglia ipotizzata
- IoC estratti e regola proposta
- Errori dell'assistente LLM documentati

Lecture e autovalutazione

Lecture obbligatorie

- Sikorski & Honig, *Practical Malware Analysis*
- Ucci, Aniello, Baldoni, *Survey of machine learning techniques for malware analysis*, Computers & Security, 2019
- Documentazione CAPEv2

Domande di autovalutazione

1. Quali feature statiche sono più robuste all'offuscamento?
2. Perché un verdetto ML va corroborato con l'analisi dinamica?
3. Quali tecniche di evasione inficiano una sandbox?
4. Quali errori tipici commette un LLM nello spiegare codice decompilato?
5. Come si passa dall'analisi del campione alle azioni di contenimento?

FINE DELLA LEZIONE · MODULO 11

Grazie

Malware analysis con AI · Corso «Incident Response
potenziata dall'AI»

Domande e materiali: ricevimento del docente. Profilo e contatti:
vincenzocalabro.it



Vincenzo Calabrò

CYBER SECURITY · DIGITAL FORENSICS



VINCENZOCALABRO.IT