

```
root@hacklab:~/parte-01/lezione-09
```

ETHICAL HACKING POTENZIATO DALL'IA // PARTE I

LEZIONE 09

Reverse Engineering e Malware Analysis

Analisi statica e dinamica del software e del malware in ambiente isolato.

docente **Vincenzo Calabro** fase Analisi difensiva: Reverse Engineering e Malware Analysis

```
root@hacklab:~/lezione-09/agenda $
```

LEZIONE 09 / AGENDA

In questa lezione

// PERCORSO

- > Assembly e formati eseguibili
- > Analisi statica e dinamica
- > Packing e anti-analysis
- > YARA e indicatori di compromissione
- > IA per triage dei campioni

metodo obiettivi -> teoria -> strumenti -> laboratorio -> caso di studio -> etica -> verifica

```
root@hacklab:~/lezione-09/obiettivi $
```

LEZIONE 09 / OBIETTIVI

Obiettivi di apprendimento

// AL TERMINE SAPRAI (TASSONOMIA DI BLOOM)

- > Comprendere – fondamenti di RE e malware analysis
- > Applicare – analisi statica e dinamica isolata
- > Analizzare – comportamento e indicatori (IOC)
- > Valutare – tecniche anti-analysis
- > Creare – una regola YARA e un mini-report

```
root@hacklab:~/lezione-09/inquadramento $
```

LEZIONE 09 / INQUADRAMENTO

Inquadramento

// DI COSA PARLIAMO

Comprendere come è fatto un binario o un malware permette di anticiparne il comportamento e produrre indicatori utili alla difesa. Introduciamo analisi statica e dinamica in ambiente isolato, con attenzione al contenimento.

// PERCHE' CONTA

Analisi statica e dinamica del software e del malware in ambiente isolato.

```
root@hacklab:~/lezione-09/mappa-insieme $
```

LEZIONE 09 / MAPPA // INSIEME

Mappa d'insieme del modulo

■ Attacco/tecnica ■ Difesa ■ Norme & metodo ■ IA

FONDAMENTI

01 Assembly e formati eseguibili

02 Analisi statica e dinamica

03 Packing e tecniche anti-analisi

04 YARA, IOC e memory forensics

05 IA per triage dei campioni

APPROFONDIMENTI

06 Disassembler e riconoscimento pattern

07 Sandbox e osservazione comportamentale

08 Superare anti-debug e packing

09 Memory forensics del malware

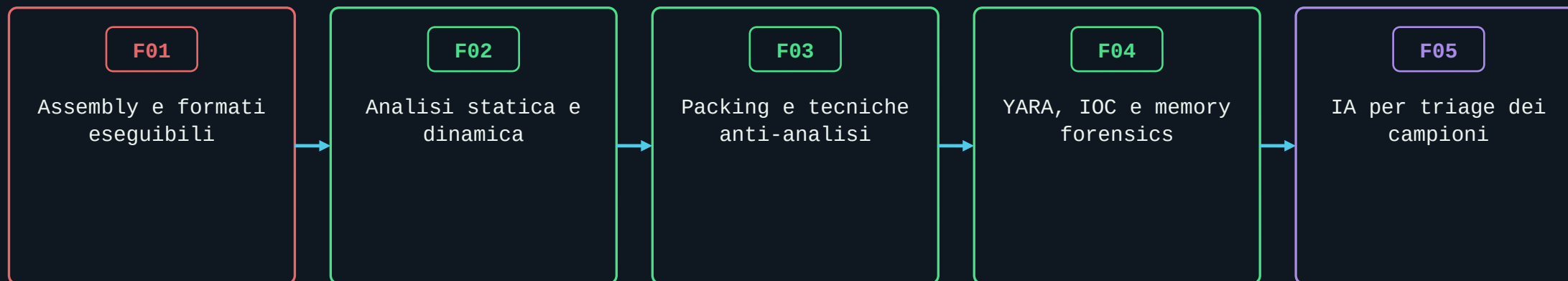
10 Classificazione e similarity

```
root@hacklab:~/lezione-09/flusso-1 $
```

LEZIONE 09 / FLUSSO // FONDAMENTI

Sequenza di fasi – Fondamenti

■ Attacco/tecnica ■ Difesa ■ Norme & metodo ■ IA



> Le fasi si susseguono: l'esito di ciascuna alimenta la successiva.

```
root@hacklab:~/lezione-09/concetto-1 $
```

LEZIONE 09 / CONCETTO 1/10

[Attacco/tecnica]

Assembly e formati eseguibili

// SPIEGAZIONE

Comprendere un binario richiede basi di assembly x86/x64 e dei formati PE su Windows ed ELF su Linux: sezioni, import, entry point. E l'alfabeto con cui si legge cosa fa davvero un programma, oltre al suo nome.

// ESEMPIO

> La tabella degli import di un PE rivela chiamate a funzioni di rete insospettate.

```
root@hacklab:~/lezione-09/concetto-2 $
```

LEZIONE 09 / CONCETTO 2/10

[Difesa]

Analisi statica e dinamica

// SPIEGAZIONE

L'analisi statica studia il codice senza eseguirlo con Ghidra, IDA o Radare2; quella dinamica lo osserva in esecuzione controllata con x64dbg o la sandbox Cuckoo. Le due si completano: la statica spiega il come, la dinamica mostra cosa accade.

// ESEMPIO

> Ghidra decompila la routine; la sandbox conferma la connessione C2 a runtime.

```
root@hacklab:~/lezione-09/concetto-3 $
```

LEZIONE 09 / CONCETTO 3/10

[Difesa]

Packing e tecniche anti-analisi

// SPIEGAZIONE

Il malware spesso è compresso o cifrato (packing) e include trucchi anti-debug e anti-VM per non rivelarsi in analisi. Riconoscerli e superarli, con unpacking e patching dei controlli, è parte del lavoro dell'analista.

// ESEMPIO

> Il campione termina se rileva una VM; si neutralizza il controllo per osservarne il comportamento.

```
root@hacklab:~/lezione-09/concetto-4 $
```

LEZIONE 09 / CONCETTO 4/10

[Difesa]

YARA, IOC e memory forensics

// SPIEGAZIONE

Le regole YARA descrivono pattern per riconoscere famiglie di malware; gli IOC come hash, domini e chiavi alimentano la difesa. La memory forensics con Volatility estrae processi e artefatti dalla RAM, dove il malware spesso vive senza toccare il disco.

// ESEMPIO

```
$ vol.py -f dump.mem windows.malfind (individua codice iniettato in memoria)
```

```
root@hacklab:~/lezione-09/concetto-5 $
```

LEZIONE 09 / CONCETTO 5/10

[IA]

IA per triage dei campioni

// SPIEGAZIONE

Di fronte a migliaia di campioni, il machine learning aiuta a raggrupparli e classificarli per similarità, accelerando il triage. Resta un supporto: la conferma sui campioni critici richiede l'analisi manuale di un esperto.

// ESEMPIO

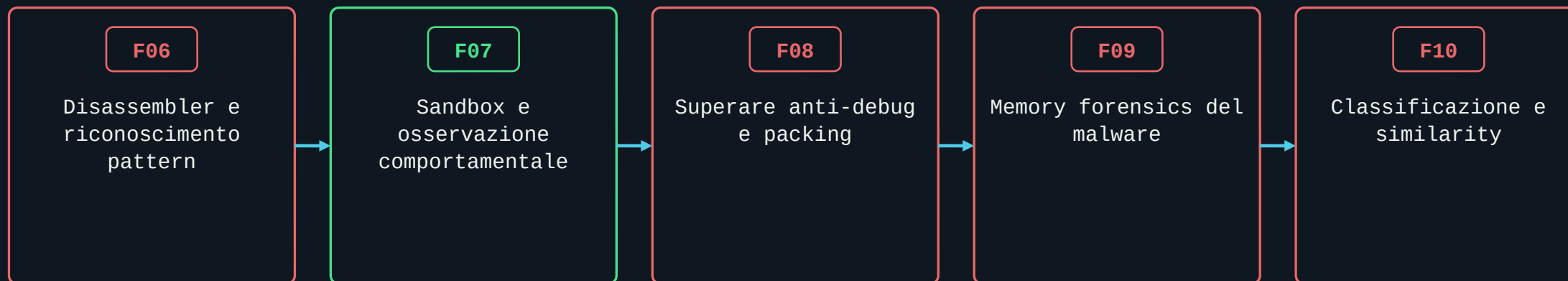
> Un modello raggruppa varianti della stessa famiglia; l'analista verifica quelle anomale.

```
root@hacklab:~/lezione-09/flusso-2 $
```

LEZIONE 09 / FLUSSO // APPROFONDIMENTI

Sequenza di fasi – Approfondimenti

■ Attacco/tecnica ■ Difesa ■ Norme & metodo ■ IA



> Le fasi si susseguono: l'esito di ciascuna alimenta la successiva.

```
root@hacklab:~/lezione-09/concetto-6 $
```

LEZIONE 09 / CONCETTO 6/10

[Attacco/tecnica]

Disassembler e riconoscimento pattern

// SPIEGAZIONE

Disassembler e decompiler traducono il binario in assembly e pseudo-codice leggibile; riconoscere pattern ricorrenti come loop di cifratura o routine di persistenza accelera la comprensione. L'esperienza nel vedere questi schemi distingue l'analista.

// ESEMPIO

> Una routine XOR su un buffer segnala la decifratura di una stringa o di un payload nascosto.

```
root@hacklab:~/lezione-09/concetto-7 $
```

LEZIONE 09 / CONCETTO 7/10

[Difesa]

Sandbox e osservazione comportamentale

// SPIEGAZIONE

In sandbox si osserva cosa il campione crea, modifica e contatta: file scritti, chiavi di registro, connessioni di rete. Questo profilo comportamentale produce IOC utili alla difesa anche senza decompilare tutto il codice.

// ESEMPIO

> Il campione crea una chiave Run per la persistenza e contatta un dominio C2.

```
root@hacklab:~/lezione-09/concetto-8 $
```

LEZIONE 09 / CONCETTO 8/10

[Attacco/tecnica]

Superare anti-debug e packing

// SPIEGAZIONE

Il malware controlla la presenza di debugger e macchine virtuali e si comprime per nascondersi. L'unpacking recupera il codice reale in memoria dopo l'auto-estrazione; superare i controlli richiede patching mirato o breakpoint nel punto giusto.

// ESEMPIO

> Dump del processo dopo l'unpacking in memoria per analizzare il codice reale.

```
root@hacklab:~/lezione-09/concetto-9 $
```

LEZIONE 09 / CONCETTO 9/10

[Attacco/tecnica]

Memory forensics del malware

// SPIEGAZIONE

Molte minacce moderne operano solo in RAM. Analizzare un dump di memoria con Volatility rivela processi nascosti, codice iniettato e connessioni attive che su disco non lascerebbero traccia: elemento chiave dell'analisi fileless.

// ESEMPIO

```
$ vol.py -f dump.mem windows.netscan (connessioni dai processi in memoria)
```

```
root@hacklab:~/lezione-09/concetto-10 $
```

LEZIONE 09 / CONCETTO 10/10

[Attacco/tecnica]

Classificazione e similarity

// SPIEGAZIONE

Confrontare campioni per similarità con fuzzy hashing e feature statiche collega varianti alla stessa famiglia o attore, utile per attribuzione e priorità. Un singolo IOC isolato dice poco; una famiglia riconosciuta orienta la risposta.

// ESEMPIO

> L'import hash (imphash) identico collega due binari diversi alla stessa origine.

```
root@hacklab:~/lezione-09/strumenti $
```

LEZIONE 09 / STRUMENTI

Strumenti della lezione

// TOOLKIT

Ghidra

IDA Pro

Radare2

Cutter

x64dbg

YARA

Cuckoo

Volatility

// NOTA OPERATIVA

> Ogni strumento si usa in ambiente virtualizzato e isolato, solo su bersagli di proprietà o autorizzati. Output e artefatti vanno protetti e, a fine ingaggio, gestiti o distrutti.

```
root@hacklab:~/lezione-09/lab-1 $
```

LEZIONE 09 / LABORATORIO 1

Laboratorio 1

\$ Analisi statica + dinamica (Ghidra + sandbox)

// PROCEDURA

[1] Apri un binario in Ghidra

[2] individua le funzioni e correla il comportamento in sandbox

[!] Verifica: Quando la dinamica svela ciò che la statica nasconde?

```
root@hacklab:~/lezione-09/lab-2 $
```

LEZIONE 09 / LABORATORIO 2

Laboratorio 2

\$ Detection con YARA ed estrazione di IOC

// PROCEDURA

[1] Scrivi e testa una regola YARA robusta ed estrai gli IOC

[!] Verifica: Cosa rende una regola YARA robusta ma non generica?

```
root@hacklab:~/lezione-09/caso-di-studio $
```

LEZIONE 09 / CASO DI STUDIO

Caso di studio

// SCENARIO

> Struttura di un malware storico (Stuxnet o un ransomware): estrazione di IOC e TTP.

// ANALISI RICHIESTA

- > Ricostruisci la dinamica dei fatti con metodo e tracciabilità
- > Individua errori, cause radice e punti di decisione critici
- > Proponi contromisure, controlli e lezioni apprese

```
root@hacklab:~/lezione-09/etica $
```

LEZIONE 09 / ETICA & DIRITTO

Nota etico-legale

// PRINCIPI INDEROGABILI

> autorizzazione scritta scope definito ambiente isolato dati protetti responsible disclosure

// SPECIFICO DELLA LEZIONE

[!] Solo ambiente isolato e non connesso; mai eseguire campioni reali fuori dalla sandbox o ridistribuirli.

```
root@hacklab:~/lezione-09/ricapitolazione $
```

LEZIONE 09 / RICAPITOLAZIONE

Ricapitolazione

// PUNTI CHIAVE

- > Fondamenti di RE e malware analysis
- > Analisi statica e dinamica isolata
- > Comportamento e indicatori (IOC)
- > Tecniche anti-analysis
- > Una regola YARA e un mini-report

```
root@hacklab:~/lezione-09/verifica $
```

LEZIONE 09 / AUTOVALUTAZIONE

Autovalutazione

// DOMANDE DI VERIFICA

- ? Quando la dinamica svela ciò che la statica nasconde?
- ? Cosa rende una regola YARA robusta ma non generica?
- ? Quali segnali rivelano il packing di un campione?

// VALUTAZIONE

Mini-report con comportamento, IOC e regola YARA.

// LETTURE

Sikorski & Honig, Practical Malware Analysis; doc. Ghidra/YARA.

lezione 09 completata

\$ next

Reverse Engineering e Malware Analysis

[!] Le competenze di questa lezione si impiegano esclusivamente in contesti legali, autorizzati e deontologicamente corretti, entro lo scope e con autorizzazione scritta.

root@hacklab:~# _

```
root@hacklab:~/contatti $
```

CONTATTI / RISORSE

Restiamo in contatto

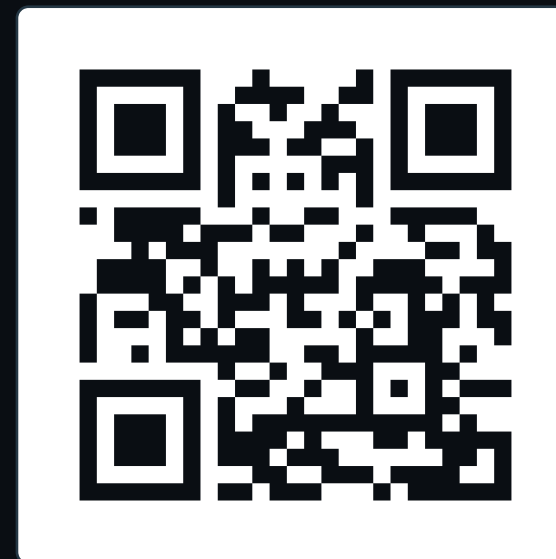


Vincenzo Calabro

cyber security / digital forensics /
docenza e divulgazione

// SCANSIONA O VISITA

> vincenzocalabro.it



Inquadra il QR con la fotocamera
per aprire il sito.