

W8D4 – Esame del modulo 2 di Maria Zanchetta

Esercizio obbligatorio: Gameshell.

Dopo aver scaricato Gameshell, ho completato tutti i livelli fino al livello 31 compreso e qui riporto gli screenshot del livello 31 completato. Nel repository con la consegna dell'esame ho inserito anche il file di salvataggio di Gameshell, che all'apertura mostra la missione del livello 32.

```
[mission 31] $ gsh goal

+-----+
| Mission goal                                     |
+-----+
| To get better in the magical art, one needs to know mental math. |
| Get ready, because Merlin is about to test your speed with products. |
| Run the command ``gsh check`` to start. |
| Remark |
+-----+
| There now is a time constraint. |
| Hint |
+-----+
| The library is rumored to contain some mathematics books and hidden |
| volumes. |
+-----+
| Useful commands |
+-----+
| COMMAND < FILE |
| Replace the command's standard input by a file. |
| Instead of reading lines from the keyboard device, the command will |
| read lines from the file. |
+-----+

~
[mission 31] $ pwd
/home/kali/Downloads/gameshell/World

~
[mission 31] $ cd ~/Castle
```

```

~
[mission 31] $ pwd
/home/kali/Downloads/gameshell/World

~
[mission 31] $ cd ~/Castle

~/Castle
[mission 31] $ ls
Cellar/  forest/  Great_hall/  Main_building/  Main_tower/  Observatory/

~/Castle
[mission 31] $ cd ~/Castle/Main_building/

~/Castle/Main_building
[mission 31] $ ls
Library/  Throne_room/

~/Castle/Main_building
[mission 31] $ cd ./Library
bash: cd: ./Library: No such file or directory

~/Castle/Main_building
[mission 31] $ ./Library
bash: ./Library: Is a directory

~/Castle/Main_building
[mission 31] $ cd ~/Castle/Main_building/Library/

```

```

~/Castle/Main_building/Library
[mission 31] $ ls
Greek_Latin_and_other_modern_languages  Mathematics_101  Merlin_s_office/

~/Castle/Main_building/Library
[mission 31] $ gsh check < Mathematics_101
93 * 75 = ?? 10 * 86 = ?? 50 * 92 = ?? 80 * 36 = ?? 13 * 43 = ?? 92 * 100 = ?
? 94 * 3 = ?? 72 * 65 = ?? 61 * 20 = ?? 17 * 57 = ?? 23 * 85 = ?? 91 * 92 = ?
? 3 * 36 = ?? 96 * 60 = ?? 42 * 8 = ?? 58 * 22 = ?? 1 * 97 = ?? 23 * 47 = ??
2 * 37 = ?? 9 * 27 = ?? 31 * 16 = ?? 84 * 68 = ?? 31 * 54 = ?? 1 * 34 = ?? 93
* 15 = ?? 18 * 95 = ?? 70 * 6 = ?? 72 * 2 = ?? 73 * 12 = ?? 96 * 63 = ?? 43
* 61 = ?? 80 * 84 = ?? 19 * 97 = ?? 63 * 16 = ?? 88 * 55 = ?? 2 * 11 = ?? 61
* 73 = ?? 52 * 96 = ?? 6 * 44 = ?? 44 * 25 = ?? 11 * 70 = ?? 99 * 66 = ?? 3 *
50 = ?? 62 * 1 = ?? 87 * 48 = ?? 89 * 16 = ?? 37 * 9 = ?? 93 * 83 = ?? 33 *
27 = ?? 39 * 89 = ?? 46 * 28 = ?? 83 * 25 = ?? 64 * 83 = ?? 15 * 37 = ?? 75 *
56 = ?? 96 * 20 = ?? 53 * 33 = ?? 65 * 38 = ?? 83 * 59 = ?? 71 * 80 = ?? 29
* 58 = ?? 59 * 21 = ?? 18 * 35 = ?? 26 * 61 = ?? 45 * 61 = ?? 63 * 2 = ?? 5 *
85 = ?? 44 * 35 = ?? 57 * 96 = ?? 26 * 93 = ?? 24 * 38 = ?? 64 * 59 = ?? 62
* 56 = ?? 62 * 26 = ?? 57 * 97 = ?? 63 * 17 = ?? 81 * 84 = ?? 89 * 97 = ?? 34
* 44 = ?? 58 * 11 = ?? 62 * 98 = ?? 59 * 73 = ?? 19 * 67 = ?? 31 * 79 = ?? 9
0 * 66 = ?? 85 * 41 = ?? 13 * 47 = ?? 21 * 6 = ?? 10 * 52 = ?? 28 * 50 = ?? 3
0 * 84 = ?? 10 * 79 = ?? 47 * 58 = ?? 98 * 85 = ?? 60 * 46 = ?? 51 * 96 = ??
29 * 21 = ?? 89 * 2 = ?? 13 * 63 = ?? 88 * 23 = ??
Congratulations, mission 31 has been successfully completed!

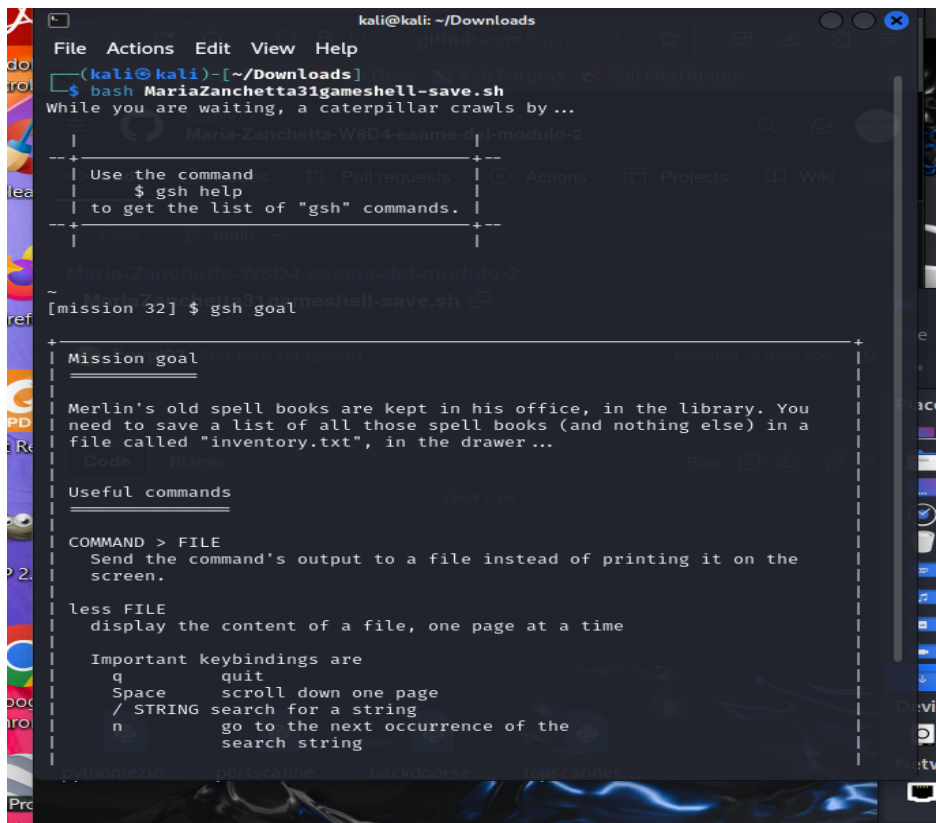
```

While you are waiting, a bat flies by...

```

| Pictures |
+-----+
| Use the command |
| $ gsh help |
| to get the list of "gsh" commands. |
+-----+
| Pictures |

```

A screenshot of a Kali Linux terminal window. The window title is 'kali@kali: ~/Downloads'. The terminal shows a script being executed: `bash MariaZanchetta31gameshell-save.sh`. The script displays a message: 'While you are waiting, a caterpillar crawls by...'. Below this, there is a decorative border containing instructions: 'Use the command \$ gsh help to get the list of "gsh" commands.' The script then prompts the user with '[mission 32] \$ gsh goal'. The user enters 'goal', and the terminal displays the following information:
Mission goal
Merlin's old spell books are kept in his office, in the library. You need to save a list of all those spell books (and nothing else) in a file called "inventory.txt", in the drawer...
Useful commands
COMMAND > FILE
Send the command's output to a file instead of printing it on the screen.
less FILE
display the content of a file, one page at a time
Important keybindings are
q quit
Space scroll down one page
/ STRING search for a string
n go to the next occurrence of the search string

Esercizio bonus: steganografia e linguaggi esoterici.

Steganografia

La steganografia è una tecnica che permette di nascondere delle informazioni all'interno di un altro messaggio o di un oggetto per evitare che vengano trovate, scrivendo quindi "in modo nascosto". Questo termine deriva dal greco antico "steganos" che significa "nascosto" o "coperto" e "graphein", ovvero "scrittura". Questa pratica è molto antica e veniva utilizzata dagli antichi Greci e dai Romani, i quali usavano la cera per nascondere dei messaggi o scrivevano con inchiostri invisibili. La steganografia oggi è molto utilizzata nel mondo della cybersecurity e può essere usata sia per scopi malevoli, sia per scopi difensivi e per la protezione dei dati. Per la protezione dei dati, la steganografia è spesso utilizzata in combinazione con la crittografia, perché da una parte la crittografia serve a rendere illeggibili le informazioni per chi non possiede la chiave per decifrare, mentre la steganografia porta ad un livello di sicurezza superiore, dato che mira a nascondere l'esistenza stessa dei dati. Combinando la steganografia con la crittografia, quindi, il messaggio che si vuole nascondere, detto payload, viene criptato e poi inserito nel file di copertura, chiamato "cover" o "container". In questo modo, le informazioni sono completamente nascoste e sono illeggibili qualora venissero trovate da persone che non possiedono la chiave per decifrarle, riuscendo quindi a proteggere informazioni particolarmente sensibili. Qualora non venissero crittografati, i dati da nascondere con la steganografia vengono comunque elaborati per renderne difficile la lettura e per consentire di decifrarli solo al vero destinatario dell'informazione che li estrarrà. Per questo motivo potrebbero essere scritti proprio con un linguaggio di programmazione esoterico, che analizzerò dopo. La steganografia può essere

applicata a diversi tipi di file, come immagini, video, file audio, testi oppure anche alla rete. La steganografia di testo prevede che le informazioni siano nascoste in un file di testo e si possono alterare il formato del testo, il carattere, sostituire delle parole, inserire il messaggio ad intervalli precisi oppure generare dei caratteri casuali tra i quali nascondere il messaggio importante. Una tecnica che può essere utilizzata è il fontcode, sviluppata dalla Columbia University, che prevede la modifica di specifici font in determinate posizioni in un documento e tramite un'applicazione si potrà vedere il messaggio nascosto inquadrando il testo alterato. Fontcode potrebbe essere utilizzato per esempio per produrre delle etichette anticontraffazione, per il trademarking, o per altri sistemi per garantire l'autenticità di un documento. La steganografia di immagini prevede l'inserimento del messaggio da proteggere in un'immagine, mentre la steganografia audio altera la sequenza binaria di un file audio per inserire il messaggio. La steganografia video permette di occultare una grande quantità di dati in un file video ed esistono due tecniche per realizzarla: da una parte i dati possono essere incorporati direttamente nel flusso di dati compresso, dall'altra l'informazione segreta può essere inserita in un video non elaborato e che verrà compresso in un secondo momento. La steganografia di rete, chiamata anche steganografia di protocollo, inserisce le informazioni in protocolli di rete come TCP, UDP o ICMP. Una delle tecniche di steganografia più comuni consiste nella modifica del colore di ogni centesimo pixel di un'immagine per corrispondere ad una lettera dell'alfabeto. Oppure è possibile utilizzare la tecnica molto diffusa della steganografia LSB, ossia Least Significant Bit. Permette di sostituire i bit meno significativi, detti anche LSB o Least Significant Bit, di uno o più canali di colore con i bit che costituiscono il messaggio nascosto, agendo sull'ultimo bit di ogni byte. In questo modo i cambiamenti sono impercettibili e vi possono essere solo delle variazioni minime grazie allo scarso peso di questi bit, riuscendo a non destare sospetti in chi vede l'immagine con il messaggio nascosto. Questo è il principio che sta alla base degli interventi di steganografia su file audio e video, per cui viene modificata solo una minima parte del file cover e il cambiamento del contenuto visivo o audio è impercettibile. Un'altra tecnica molto diffusa consiste nell'applicazione di una filigrana che riporti il copyright di un contenuto, proteggendo i diritti d'autore in modo impercettibile ed evitando la contraffazione di un contenuto tutelato dai diritti d'autore.

La steganografia è quindi una tecnica molto importante che può essere utilizzata sia per la protezione dei dati e per la sicurezza, sia per scopi malevoli da criminali informatici. Tra gli scopi positivi per i quali può essere usata la steganografia si può citare la necessità di aggirare meccanismi di censura, di proteggere i diritti d'autore con la filigrana digitale evitando riproduzioni non autorizzate e di difendere informazioni segrete scambiate tra istituzioni o forze dell'ordine. La steganografia difensiva viene utilizzata spesso per la protezione dei dati sensibili e delle comunicazioni segrete e in alcuni casi consiste nella creazione di informazioni false inserite in file digitali che possono trarre in inganno dei malintenzionati, che penseranno che le informazioni nascoste siano vere. Oltre alla filigrana, la steganografia può essere applicata anche nel digital watermarking. Questa tecnica consiste nell'inserire un identificatore unico in media digitali, che consente al solo autore di dimostrare la proprietà intellettuale e di difendersi da riproduzioni non autorizzate.

L'identificatore è nascosto, per cui non altera il contenuto audio, video o di immagine ma è molto difficile da rimuovere e per questo viene utilizzato per contrastare la pirateria nell'industria musicale o cinematografica. La steganografia può essere utilizzata anche per scopi malevoli, per esempio per nascondere un malware o un ransomware in un file apparentemente innocuo, che però contiene un codice malevolo. Molti ransomware sono stati diffusi con la steganografia, che torna utile anche per l'esfiltrazione dei dati, ossia il trasferimento di dati che vengono nascosti in comunicazioni legittime per passare inosservati mentre sottraggono dati dalla vittima dell'attacco. È facile nascondere payload dannosi in file multimediali come le immagini, perché le immagini hanno molti dati ridondanti che possono essere facilmente manipolati passando inosservati. Una delle tecniche di steganografia che può essere usata facilmente per un attacco di questo tipo è la tecnica "stegosploit", che contiene in sé i termini "steganography" ed "exploit". È un attacco che utilizza un exploit basato sulla steganografia delle immagini, nel quale si inseriscono in un'immagine dei pezzi di codice HTML/Javascript, per cui si vede un'immagine ma un browser legge un codice javascript valido. È possibile utilizzare questa tecnica per iniettare del codice in immagini in formato .bitmap, .gif e .png, anche se ci sono delle differenze operative per giungere al risultato finale in base al tipo di immagine. Gli attacchi basati sulla steganografia possono avere effetti potenzialmente molto gravi, basti pensare all'attacco con il trojan bancario Ursnif, che si è propagato tramite un'immagine apparentemente innocua in formato .png ma che al suo interno conteneva lo script del codice malevolo del trojan. La steganografia viene utilizzata anche per nascondere comandi malevoli nelle pagine web e per campagne di malvertising, nascondendo codice dannoso ad esempio in un banner su un sito, che porta l'utente che clicca sul banner in una landing page di un kit di exploit e che fa scaricare ed eseguire del codice malevolo. Per esempio, nel 2020 è stato rilevato un attacco con malware di skimming nelle pagine di checkout di vari siti di e-commerce, realizzato con un payload dannoso in immagini SVG dei loghi delle aziende e con un decoder nascosto in altre parti dei siti. Il payload era inserito correttamente nella sintassi di elementi SVG e per questo non veniva rilevato come una minaccia. Sempre nel 2020 è stato nascosto un malware in un aggiornamento software legittimo di SolarWinds, riuscendo a violare aziende come Microsoft ed alcune agenzie governative americane, mentre molte aziende hanno subito dei danni a causa Mimikatz. Questo malware sottraeva le password di Windows e veniva scaricato tramite uno script nascosto con la steganografia in immagini scaricate legittimamente da piattaforme come Imgur e che infettavano un documento Excel. Ci sono altri tipi di attacchi basati sulla steganografia, che si classificano in base a ciò che l'attaccante è riuscito a scoprire. Gli attacchi di tipo stego-only si verificano quando l'attaccante ha intercettato il frammento stego ed è riuscito ad analizzarlo e questa è la tipologia di attacchi che si verifica più frequentemente. Il frammento stego è quello che contiene il messaggio segreto. Lo stego-attack avviene quando una stessa cover o container è stata utilizzata più volte per nascondere dei dati e l'attaccante ha ottenuto un frammento stego diverso ma originato dalla stessa cover. Il cover-stego-attack si verifica quando l'attaccante ha intercettato un frammento stego e sa quale cover è stata usata per crearlo, mentre con il cover-emb-stego-attack l'attaccante ha il frammento stego, conosce la cover usata e il

messaggio segreto nascosto tramite steganografia. Un attaccante può anche modificare i frammenti stego, sottraendo il messaggio nascosto, modificandolo o inserendo un codice malevolo. Per difendersi da possibili attacchi che utilizzano la steganografia si possono adottare degli accorgimenti come diffidare da immagini di dimensioni insolitamente elevate, proteggere gli endpoint e mantenere aggiornati i sistemi. Ci sono vari software per la steganografia, come Hide'N'Send, OpenStego, Steghide, SSuite Píscel e descriverò le particolarità e le funzioni di alcuni di questi software. Hide'N'Send è utile per operare sulle immagini in formato JPEG usando l'algoritmo di steganografia F5 e la tecnica del least significant bit. OpenStego, tra tutte le altre funzioni, consente di apporre un watermark dando come risultato un file PNG. Steghide è un software open source che opera su immagini BMP tramite tecniche di steganografia LSB. OutGuess è un altro software open source che opera su file JPEG e PNG con tecniche di manipolazione di coefficienti nel dominio trasformato attraverso la DCT. Steganos Security Suite è invece un software commerciale pensato per proteggere documenti e password ed S-Tools si avvale anche di algoritmi di crittografia per la cifratura dei dati che verranno poi inseriti tramite steganografia in un testo, in un'audio o in un'immagine. Si definisce steganalisi la tecnica che consente di individuare la presenza o meno di un messaggio nascosto tramite steganografia, utilizzando ad esempio StegExpose, StegAlyze o altri strumenti come visualizzatori hex per individuare delle anomalie legate alla steganografia.

La steganografia è dunque una tecnica molto importante da conoscere per chi opera nella cybersecurity. Permette infatti di proteggere informazioni sensibili nascondendo l'esistenza dei dati ed inserendoli in un contenitore insospettabile e consente di aggirare la censura e di difendere i diritti d'autore con la filigrana o il digital watermarking. È importante conoscere la steganografia anche perché viene usata spesso per organizzare degli attacchi ed è bene esserne consapevoli e prendere delle misure per difendersi.

Esempio di steganografia.

Ho provato ad applicare la steganografia per nascondere un messaggio potenzialmente importantissimo, come una password per un server governativo, in un'immagine comune e insospettabile. Ho utilizzato [Steganography Online \(stylesuxx.github.io\)](https://stylesuxx.github.io) per fare un piccolo esercizio di steganografia. Il messaggio che ho inserito è il seguente: la password per accedere al server governativo è ABC123456. Poi servirà scansionare il badge e ottenere un codice OTP dal telefono governativo. Dopo la scansione biometrica si entra nel server.

Questa è l'immagine normale che ho inserito:



Mentre questa è l'immagine con il messaggio nascosto. Non si nota nessun cambiamento rispetto all'immagine senza messaggio nascosto ed è un'immagine insospettabile: non ci si aspetta che contenga un messaggio importante e non si sospetta eventualmente che possa contenere un codice malevolo.



Linguaggi di programmazione esoterici.

I linguaggi di programmazione esoterici sono linguaggi di programmazione particolarmente complessi e volutamente poco chiari, che possono essere usati per molteplici scopi, come testare i limiti della programmazione su un computer, proteggere delle informazioni sensibili, comprendere meglio il funzionamento di un calcolatore oppure dimostrare come la programmazione possa avere un lato divertente, artistico e creativo. Il primo linguaggio esoterico è stato INTERCAL, scritto nel 1972 da James M. Lyon e Don Woods, per creare un linguaggio di programmazione completamente diverso da quelli esistenti, nel quale vengono ripresi e modificati alcuni dei tratti tipici di altri linguaggi. Un altro linguaggio esoterico importante è FALSE, creato da Wouter van Oortmerssen ispirandosi al concetto di stack ed utilizzando una sintassi concisa e di difficile lettura. FALSE ha ispirato il linguaggio Brainfuck creato da Urban Müller nel 1993 e che approfondirò in un secondo momento con un esempio pratico. Brainfuck è un linguaggio di programmazione esoterico particolarmente interessante, perché è minimale ed ha un alto livello di offuscamento del codice, il che lo rende interessante come linguaggio di programmazione per la sicurezza dei dati. L'obiettivo di Urban Müller era quello di sviluppare un linguaggio adatto ad una macchina di Turing che utilizzasse il più piccolo compilatore possibile, ancora più piccolo del compilatore di FALSE che occupa 1024 byte. Brainfuck è un linguaggio Turing-completo, che può essere utilizzato per implementare qualunque algoritmo eseguibile con una macchina di Turing e si basa su un array di byte inizializzato a zero, un puntatore all'array inizializzato per puntare al primo byte dell'array e due stream di byte per l'input e l'output. Le istruzioni del linguaggio sono solamente 8 e i caratteri sono i seguenti. > incrementa il puntatore, < decrementa il puntatore, + incrementa il byte indirizzato dal puntatore, - decrementa il byte indirizzato dal puntatore, . è l'output dal byte indirizzato dal puntatore, , è l'input al byte indirizzato dal puntatore, [passa in avanti all'istruzione dopo il corrispondente] se il byte indirizzato dal puntatore è zero, infine] passa indietro all'istruzione dopo il corrispondente [se il byte indirizzato dal puntatore non è zero. I codici in Brainfuck possono essere anche codificati in C, riprendendo il concetto di puntatore, per cui > equivale a ++ptr; < è -- ptr; + è ++(*ptr); - equivale a --(*ptr); . è putchar(*ptr); , è *ptr = getchar(); [è while (*ptr) { e] equivale a }. Questo linguaggio di programmazione è minimalista e particolarmente difficile da leggere, portando ai massimi l'offuscamento del codice ed è quindi un'opzione interessante per proteggere delle informazioni sensibili o strategiche. Dal sito [Brainfuck Language - Online Decoder, Translator, Interpreter \(dcode.fr\)](#) ho provato a realizzare un codice in Brainfuck per rendere molto difficile la lettura di un'ipotetica informazione militare. Il messaggio è "l'offensiva inizierà lunedì prossimo, attaccando la Russia con uno sciame di droni" e questo è il codice Brainfuck

[illegible]

[illegible]

I linguaggi di programmazione esoterici possono essere utilizzati per migliorare la sicurezza dei dati proprio perché sono volutamente molto complessi ed estremamente difficili da leggere. Grazie all'offuscamento del codice, il codice sorgente scritto con questi linguaggi è particolarmente difficile da analizzare per un attaccante in cerca delle vulnerabilità da

sfruttare, impedendogli di trovare dei punti deboli da attaccare e di capire a fondo il funzionamento del codice. Possono anche essere usati per svolgere dei test di sicurezza, verificando ad esempio se un sistema riesce a gestire un codice scritto in questi linguaggi senza avere dei comportamenti strani, testando i limiti del sistema stesso. Proprio in virtù della complessità di questi linguaggi, la discrezione delle informazioni è mantenuta, perché qualora un malintenzionato venisse in possesso delle informazioni segrete, molto probabilmente non riuscirebbe a decifrarle a causa dell'alto livello di offuscamento del codice e della scarsa leggibilità.

Combinare la steganografia con i linguaggi di programmazione esoterici protegge i dati in maniera significativa e ne garantisce la discrezione, perché da una parte la steganografia nasconde l'esistenza di questi dati occultandoli in un comune file contenitore insospettabile, mentre i linguaggi di programmazione esoterici rendono estremamente difficile la lettura delle informazioni e l'analisi dei codici scritti con questi linguaggi qualora venissero trovati. Si possono implementare anche altre misure aggiuntive per la sicurezza delle informazioni in aggiunta all'utilizzo della steganografia e dei linguaggi esoterici, ma l'utilizzo attento della steganografia per occultare le informazioni e la scelta attenta di un linguaggio esoterico complesso e ben strutturato conferiscono un livello di sicurezza avanzato alle informazioni.