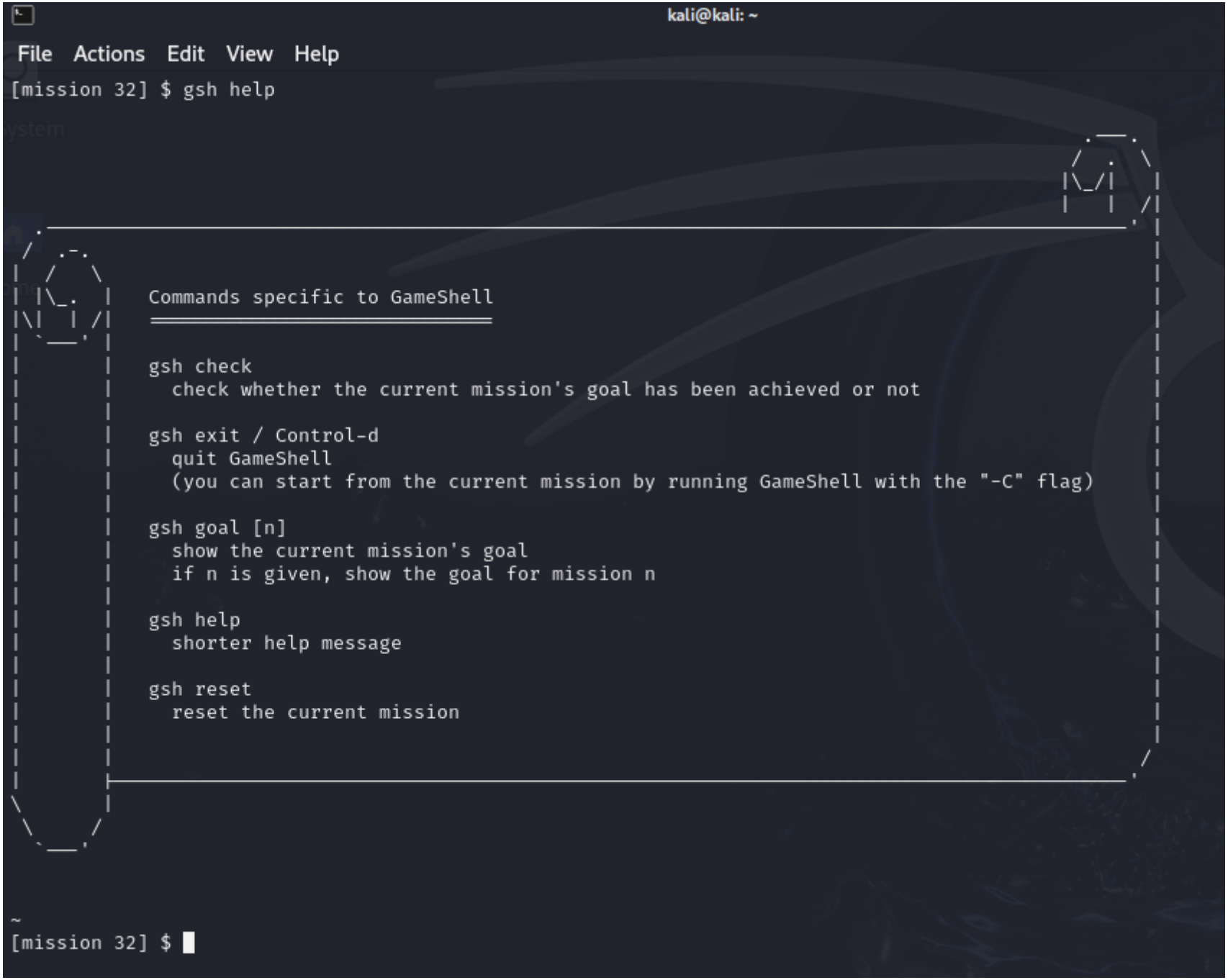
W8D4 - Pratica

Gioco gameshell


kali@kali: ~
File Actions Edit View Help
[mission 32] $ gsh help
Commands specific to GameShell
gsh check
check whether the current mission's goal has been achieved or not
gsh exit / Control-d
quit GameShell
(you can start from the current mission by running GameShell with the "-C" flag)
gsh goal [n]
show the current mission's goal
if n is given, show the goal for mission n
gsh help
shorter help message
gsh reset
reset the current mission
[mission 32] $

Steganografia

Per nascondere il significato di un messaggio che può essere letto da chi non dovrebbe si usa la crittografia (dal greco kryptós, “nascosto”, e graphía, “scrittura”**), la steganografia (dal greco steganós, «coperto», e graphía, «scrittura») invece è l'arte e la scienza di nascondere direttamente il messaggio (che sia crittografato o meno). Lo scopo della crittografia è l'occultazione del significato di un messaggio, quello della steganografia è l'occultazione del messaggio stesso.
In informatica i messaggi e le informazioni sono codificati in file.

Tipi di steganografia

Per ogni tipo di file ci possono essere un numero indefinito di modi per steganografarli, in base alle conoscenze e creatività dello steganografo. Ecco alcuni esempi di come steganografare dei tipi di file:

Immagini

La più semplice steganografia di un file può essere fatta con la modifica di una immagine. L'occhio umano non riesce a distinguere piccole variazioni di colore dei pixel di una immagine, modificando i bit meno significativi dei pixel della immagine (Least Significant Bits, LSB) possiamo codificare in essi l'informazione del file che vogliamo nascondere.

Audio

Come per le immagini, l'orecchio umano non riesce a riconoscere impercettibili variazioni delle frequenze sonore e possono essere modificate per codificarci informazioni

Video

Un video solitamente è molto più pesante di una immagine e di un audio (visto che è fatto di entrambi). è possibile nasconderci file più pesanti usando tecniche miste delle prime due tipologie accennate.

Testo

Si può modificare un testo in chiaro grazie alla spaziatura delle parole(con margini o spazi), cambio di font o colore oppure indentazione che nascondono un testo codificato.

Esempi di steganografia

Col sito Steganography Online (stylesuxx.github.io) sono riuscito a steganografare una immagine. Questa è quella originale:

Questa è quella modificata:

Questo è il testo in chiaro, codificato in binario che è stato steganografato:

Encode message

To encode a message into an image, choose the image you want to use, enter your text and hit the **Encode** button.
Save the last image, it will contain your hidden message.
Remember, the more text you want to hide, the larger the image has to be. In case you chose an image that is too small to hold your message you will be informed.

Neither the image nor the message you hide will be at any moment transmitted over the web, all the magic happens within your browser.

Scegli il file 001.png

Bulbasaur è un Pokémon di doppio tipo Erba/Veleno introdotto in prima generazione.

Si evolve in Ivysaur a partire dal livello 16 e quindi in Venusaur a partire dal livello 32.

Insieme a Charmander e Squirtle, Bulbasaur è uno dei tre Pokémon iniziali di Kanto disponibili in Pokémon Rosso e Verde, Pokémon Blu, Pokémon Rosso Fuoco e Verde Foglia.

Encode

Binary representation of your message

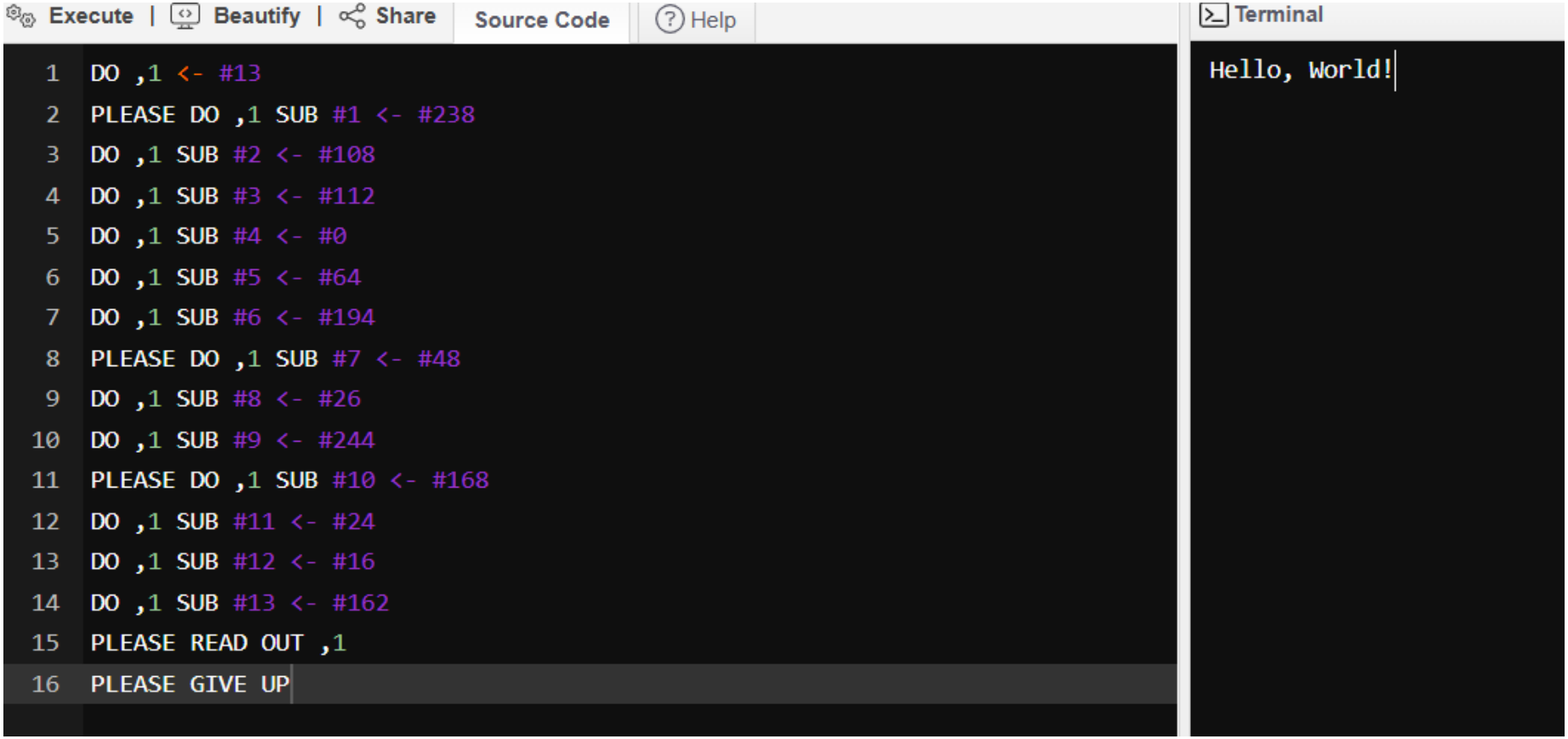
0100001001110101011011000110001001100001011100110110000101110101011100100010000011101000001000000111010101101110001000000101000001101111011010111110100101101101011
0111101101110001000000110010001101001001000000110010001101111011100000111000001101001011011110010000001110100011010010111000001101111001000000100010101110010011000
1001100001001011110101011001100101011011000110010101101110011011110010000001101001011011100111010001110010011011110110010001101111011101000111010001101111001000001
1010010110111000100000011100000111001001101001011011010110000100100000011001110110010101101110011001010111001001100001011110100111010010110111101101110011001010010
1100000101000001010010100110110100100100000011001010111011001101111011011000111011001100101001000000110100101101110001000000100100101110110011110010111001101100000
1011101010111001000100000011000010010000001110000011000010111001001110100011010010111001001100101001000000110010001100001011011000010000001101100011010010111011000
1100101011011000110110001101111001000000110001001101100010000001100101001000000111000010111010101101001011011100110010001101001001000000110100101101110001000000010
1011001100101011101100111010101110011011000010111010101110010001000000110000100100000011100000110000101110010011101000110100101110010011001010010000001100100011000
0010110110000100000011011000110100101110110011001010110110001101100011011110010000000110011001100100010111000001010000010100100100101101110011100110110100101100101
1011011010110010100100000011000010010000001000011011010000011000010111001001101101011000001011011100110010001100101011100100010000001100101001000000101001101110001
0111010101101001011100100111010001101100011001010010110000100000010000100111010101101100011000010011000010111001101100001011101010111001000100000011101000001000000
1101010110111001101111001000000110010001100101011010010010000001110100011100100110010100100000010100000110111101101011111010010110110101101111011011100010000001101
0010110111001101001011110100111010010110000101101100011010010010000001100100011010010010000001001011011000010110111001110100011011110010000001100100011010010111001
1011100000110111101101110011010010110001001101001011011000110100100100000011010010110111000010000001010000011011110110101111101001011011010110111101110010000001
0100100110111101110011011100110111101111001000000110010100100000010101100110010101110010011001000110010100100000100011001101111011001110110110001101001011011110
0100000010001100111010101101111011000110110111100100000011001010010000001010110011001010111001001100100011001010010000001000110011011110110011101101100011010010110
000100101110

Original

Un altro esempio è la conversione di una immagine in un file audio con **Slow-scan television** (**SSTV**).
La codifica SSTV funziona convertendo le informazioni visive di un'immagine in segnali audio. L'immagine viene divisa in righe, poi ogni pixel di una riga viene codificato come una frequenza audio. Frequenze diverse hanno intensità e colori diversi. Le righe dell'immagine vengono trasmesse una alla volta come un continuo segnale audio. Ci sono segnali per indicare al ricevente quando finisce una riga per poi decodificarla. E' possibile quindi nascondere in un file audio (magari sommandolo a una canzone un un altro file, come rumore di fondo per poi risepararlo) un file di immagine.
Nel videogioco Portal, come ARG, viene usato questo sistema per nasconderci delle immagini:
Portal Update 3/1/2010 (transmission received) DECODED (youtube.com)

Linguaggi esoterici

I linguaggi di programmazione esoterici (dal greco antico ἐσωτερικός, esōterikós, a sua volta da ἐσώτερος, esṓteros, "interiore") sono dei linguaggi inventati, volutamente complicati e poco pratici, popolari fra gli hacker. Li accomuna il fatto di essere Turing-equivalenti (quindi in teoria ogni algoritmo può essere convertito in un linguaggio esoterico).
Questi linguaggi hanno sia lo scopo didattico (forzare i limiti teorici per imparare il funzionamento di sintassi e i limiti delle macchine) o per divertimento (inventare il linguaggio più strano).
Il primo linguaggio esoterico è stato INTERCAL (1972), scritto da James M. Lyon e Don Woods e nato come parodia dei linguaggi già esistenti come Fortran COBOL e assembly (aveva elementi simili a loro ma era differente).
Ecco un esempio di codice INTERCAL eseguito:



Dopo INTERCAL, a seguire, sono nati molti altri linguaggi esoterici, dal Brainfuck, che ha una sintassi minimale composta solo da otto caratteri(>,<,+,- "." , "," , [,]), al Malbolge, ideato apposta per essere il più difficile da utilizzare. Ad esempio in Malbolge per eseguire la stampa di un normale "Hello, World!" il codice è questo:

```
('&%:9]!~}|z2Vxwv-,POqponl$Hjig%eB@@>}=<M:9wv6WsU2T|nm-,jcL(I&%$#"
`CB]V?Tx<uVtT`Rpo3NlF.Jh++FdbCBA@?]!~|4XzyTT43Qsqq(Lnmkj"Fhg${z@>
```

L'uso di linguaggi di difficile comprensione può essere utile per la sicurezza dei dati:
offuscare il significato in chiaro di un codice con uno più complesso può essere utile per ritardare la comprensione di come funziona il codice e quindi rallentarne la ricerca di vulnerabilità.