Introduzione alle CTF

Lezione 1

Alessandro Righi Cristiano Di Bari

Università degli Studi di Verona

3 Novembre 2023

Chi siamo?

- laurea magistrale @ UniVR
- istruttore @ CyberChallenge
- *System Developer* @ IOTINGA

Cristiano Di Bari

Alessandro Righi

- laurea magistrale @ UniVR
- istruttore @ CyberChallenge
- *Applied Research Director* @ IOTINGA

Cosa sono le CTF?

Le *Capture The Flag* (CTF) sono delle sfide in cui i partecipanti devono trovare delle *flag*, ossia delle stringhe di testo, all'interno di sistemi informatici contenenti delle vulnerabilità di sicurezza.

Esempio di flag

`CCIT{Th1s-1sY0uR-F1rst-Fl4ag}`

Una volta trovate le flag vanno, solitamente, inviate ad una piattaforma di gara, che si occupa di validarle, assegnando il punteggio della challenge nel caso siano corrette.

Perché fare CTF?

Avete mai fatto CTF?

Perché fare CTF?

Avete mai fatto CTF?

Se no... ecco alcune ragioni per iniziare:

Perché fare CTF?

Avete mai fatto CTF?

Se no... ecco alcune ragioni per iniziare:

- per divertirsi!

Perché fare CTF?

Avete mai fatto CTF?

Se no... ecco alcune ragioni per iniziare:

- per divertirsi!
- per imparare cose nuove (tante)

Perché fare CTF?

Avete mai fatto CTF?

Se no... ecco alcune ragioni per iniziare:

- per divertirsi!
- per imparare cose nuove (tante)
- per scrivere software (più) sicuro

Perché fare CTF?

Avete mai fatto CTF?

Se no... ecco alcune ragioni per iniziare:

- per divertirsi!
- per imparare cose nuove (tante)
- per scrivere software (più) sicuro
- per conoscere nuova gente, creare networking

Tipologie di CTF

Esistono due tipologie di CTF:

- *Jeopardy*: il partecipante attacca una serie di servizi malevoli e sottomette le flag ad un sistema di verifica
- *Attack-Defence (AD)*: competizione a squadre dove ogni team deve attaccare i sistemi dell'avversario, e difendere i propri

Per queste lezioni di concentreremo sul primo tipo (*Jeopardy*), di gran lunga le più diffuse, che è anche quella organizzata da *Würth Phoenix*.

Tipologie di challenge

Tipicamente le challenge che si affrontano possono essere di 4 macro categorie:

- *Binary*: è necessario ricercare vulnerabilità in un eseguibile, quali ad es. buffer overflow
- *Web*: si tratta di trovare vulnerabilità in una web app, web API, o comunque applicativo esposto in rete
- *Crypto*: è necessario decodificare un testo cifrato con un algoritmo (ovviamente vulnerabile)
- *Misc*: sono challenge che non rientrano in nessuno dei tipi precedenti, e richiedono spesso creatività per essere affrontate

Per queste lezioni ci concentreremo sulla categoria *Web*.

Iniziamo!

Path traversal

Immaginiamo di avere una pagina web che per caricare l'immagine del profilo di un utente effettua una richiesta a:

Richiesta

```
https://mysecureapp.com/assets?name=image.jpeg
```

Path traversal

Immaginiamo di avere una pagina web che per caricare l'immagine del profilo di un utente effettua una richiesta a:

Richiesta

```
https://mysecureapp.com/assets?name=image.jpeg
```

Cosa succede se modifico la richiesta in questo modo?

Richiesta alterata

```
https://mysecureapp.com/assets?name=../image.jpeg
```

Path traversal

Immaginiamo di avere una pagina web che per caricare l'immagine del profilo di un utente effettua una richiesta a:

Richiesta

```
https://mysecureapp.com/assets?name=image.jpeg
```

Cosa succede se modifico la richiesta in questo modo?

Richiesta alterata

```
https://mysecureapp.com/assets?name=../image.jpeg
```

Se il server non effettua adeguati controlli, è possibile leggere file fuori dalla *root* directory del web server!

Path traversal

Cosa consente di fare questa vulnerabilità?

Path traversal

Cosa consente di fare questa vulnerabilità?

- leggere *segreti* altrimenti non accessibili, ad es. file di configurazione quali `/etc/passwd`

Path traversal

Cosa consente di fare questa vulnerabilità?

- leggere *segreti* altrimenti non accessibili, ad es. file di configurazione quali `/etc/passwd`
- ottenere il *codice sorgente* dell'applicazione web

Path traversal

Cosa consente di fare questa vulnerabilità?

- leggere *segreti* altrimenti non accessibili, ad es. file di configurazione quali `/etc/passwd`
- ottenere il *codice sorgente* dell'applicazione web
- accedere ai dati di altri utenti, bypassando restrizioni imposte dall'applicazione web

Path traversal

Cosa consente di fare questa vulnerabilità?

- leggere *segreti* altrimenti non accessibili, ad es. file di configurazione quali `/etc/passwd`
- ottenere il *codice sorgente* dell'applicazione web
- accedere ai dati di altri utenti, bypassando restrizioni imposte dall'applicazione web

Suggerimento

È possibile aggiungere tanti `../` fino a raggiungere la directory *root*, ad esempio `../../../../../etc/passwd`

Challenge

Vediamo la prima challenge. Per queste lezioni utilizzeremo delle challenge prese dalla piattaforma di allenamento delle Olimpiadi di Cybersecurity (`https://olicyber.it`), a cui vi invitiamo ad iscrivevi.