

Introduction web vulnerability

WEB APPLICATION & API TESTING

Agenda

- ▶ Introduction
- ▶ OWASP
- ▶ Authentication Bypass
- ▶ IDOR
- ▶ File Inclusion
- ▶ SSFR
- ▶ Cross-site Scripting
- ▶ Command Injection
- ▶ SQL Injection
- ▶ Common API security problems

Web application

- ▶ Standard architecture webapp
- ▶ Intercept the traffic between a web browser/client and web server → (BurpSuite) to understand the communication
- ▶ Authentication vs Authorization

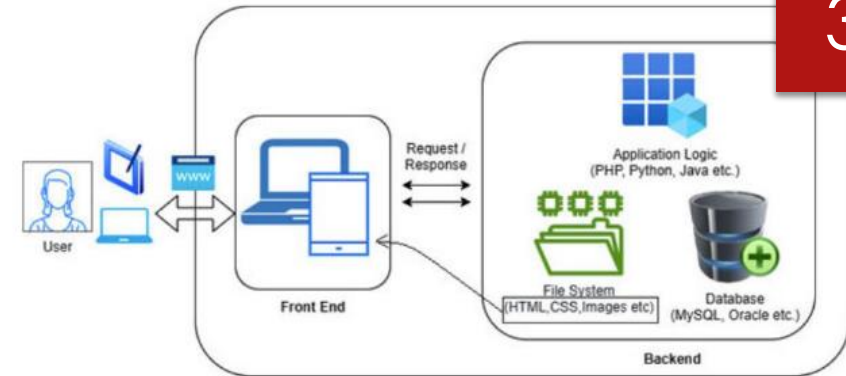


Figure 4.1: Standard web application architecture

The image shows a screenshot of a web browser's developer tools, specifically the 'Network' tab, displaying a raw HTTP request and response. The request is a POST to /doLogin with a body containing user and password information. The response is a 302 Found status with a location of login.jsp. Red circles and arrows highlight specific parts of the request and response, numbered 1 through 6. A red box labeled 'HTTP Response Body' is also visible on the right side of the response pane.

```
Request
1 POST /doLogin HTTP/1.1
2 Host: demo.testfire.net
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:94.0) Gecko/20100101 Firefox/94.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 35
9 Origin: http://demo.testfire.net
10 Connection: close
11 Referer: http://demo.testfire.net/login.jsp
12 Cookie: JSESSIONID=CFB28E659E7CE6B51A9668AFDFDC8B4E
13 Upgrade-Insecure-Requests: 1
14
15 uid=user&pass=pass&btnSubmit=Login

Response
1 HTTP/1.1 302 Found
2 Server: Apache-Coyote/1.1
3 Location: login.jsp
4 Content-Length: 0
5 Date: Sun, 05 Dec 2021 05:35:22 GMT
6 Connection: close
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

Figure 4.2: Raw HTTP request and response

Different types of web application vulnerabilities

- Insecure digest
- Insecure Direct Object References (IDOR)
- Insufficient logging and monitoring
- Insufficient session expiration
- Insufficient transport layer protection
- Lightweight Directory Access Protocol (LDAP) injection
- Missing function level access control
- Operating System (OS) command injection
- Race condition
- Remote Code Execution (RCE)
- Remote File Inclusion (RFI)
- Security misconfiguration
- Sensitive data exposure
- Session ID leakage
- SQL injection
- Invalidated redirects and forwards
- Broken access control
- Broken authentication
- Carriage Return and Line Feed (CRLF) injection
- Cipher transformation insecure
- Components with known vulnerabilities
- Cross-Origin Resource Sharing (CORS) policy
- Credentials management
- Cross-Site Request Forgery (CSRF)
- Cross-Site Scripting (XSS)
- Directory indexing
- Directory traversal
- Error handling
- Failure to restrict URL access
- HTTP response splitting
- Insecure deserialization

OWASP

- ▶ <https://owasp.org/API-Security/editions/2023/en/0x11-t10/>
- ▶ <https://owasp.org/Top10/>
- ▶ Broken Access Control
- ▶ Injection
- ▶ Insecure design
- ▶ Security Misconfiguration
- ▶ Identification and Authentication failures
- ▶ Server-Side Request Forgery

Injection – SQL Injection

- ▶ Appears when developers don't sanitize user input
- ▶ <https://site.com/profile?id=123>
- ▶ "123; DROP TABLE Users;" → SQL request run 2X
- ▶ Very high level :
 - ▶ Error-based SQLi
 - ▶ Blind SQLi

Exemple Error-based SQLi

- ▶ Supposons que l'URL du site Web permette aux utilisateurs de rechercher des articles par ID, comme suit :
`https://example.com/articles?id=1`.

- ▶ Requête SQL côté serveur :

```
SELECT * FROM Articles WHERE ArticleID = 1;
```

- ▶ Un attaquant peut entrer une entrée malveillante comme `1' OR '1'='1';--`. La requête SQL résultante sera :

```
SELECT * FROM Articles WHERE ArticleID = 1' OR '1'='1';--';
```

- ▶ Si l'application est vulnérable à l'Error-based SQL Injection, elle peut renvoyer une erreur de base de données, indiquant une vulnérabilité.

Exemple Blind SQL injection

- ▶ Supposons que le site Web ait une fonction de recherche qui renvoie une réponse différente en fonction de la vérité d'une condition, par exemple, si l'intrant est valide ou non. L'URL pourrait ressembler à ceci : `https://example.com/search?query=test`.
- ▶ `SELECT * FROM Products WHERE Name = 'test';`
- ▶ Un attaquant peut entrer une entrée malveillante comme `test' AND 1=1;--`. La requête SQL résultante sera :
- ▶ `SELECT * FROM Products WHERE Name = 'test' AND 1=1;--';`
- ▶ Si la page répond différemment (par exemple, affiche un certain message) lorsque la condition est vraie, l'attaquant peut déduire la vérité de la condition en fonction des réponses de la page, réalisant ainsi une Blind SQL Injection

Blind SQLi – Authentication bypass

9

- ▶ Little to no feedback
- ▶ Matching pair in the user table : username & Password

Blind SQLi

10

Sample Footer Text
1/11/2024

Boolean Based

- ▶ Response from our injection : true or false, yes/no , 1/0
- ▶ Confirm that injection payload was successful or not
- ▶ Possibility to enumerate a whole database structure and contents
- ▶ <https://tryhackme.com/room/sqlinjectionlm>

Boolean Based

- ▶ No visual indicator of your queries
- ▶ Based on time delay : SLEEP(x) alongside UNION statement.

- ▶ Outil : SQLMap

Remediation

As impactful as SQL Injection vulnerabilities are, developers do have a way to protect their web applications from them by following the below advice:

- ▶ **Prepared Statements (With Parameterized Queries)**
- ▶ **Input Validation**
- ▶ **Escaping User Input**
- ▶ **Enforcing least privilege**

Outils : SQLMap
Cheatsheet:

<https://portswigger.net/web-security/sql-injection/cheat-sheet>

Cross Site Scripting (XSS)

- ▶ Les attaques XSS injectent des scripts malveillants dans des sites web
- ▶ Quand : Application envoie des données non fiables au navigateur sans validation côté serveur.
- ▶ XSS: Serveur renvoi du code malveillant à un autre utilisateur.

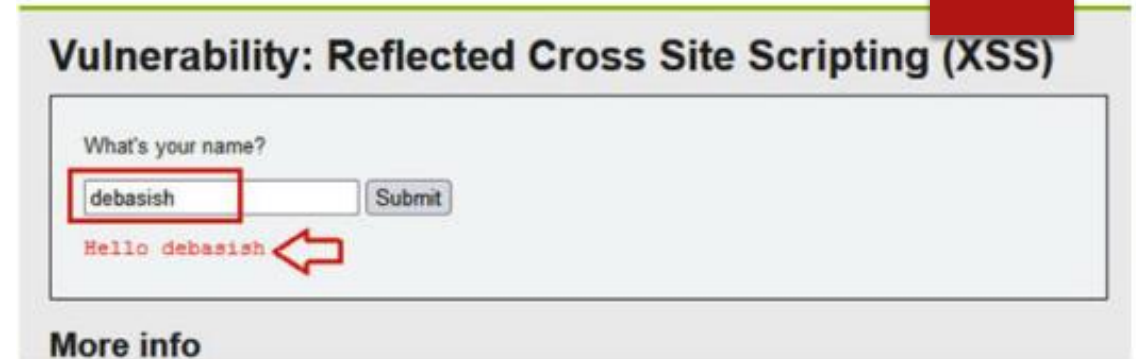


Figure 4.16: DVWA XSS

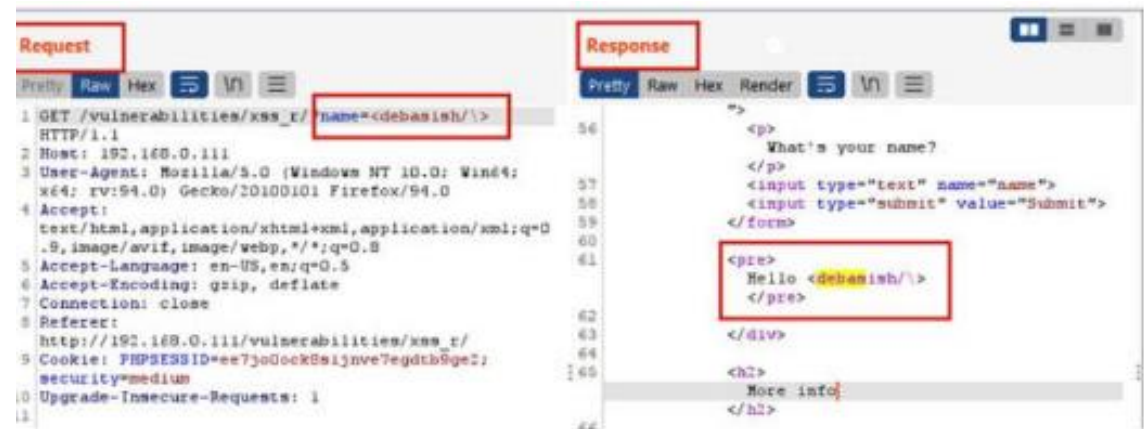


Figure 4.18: Burp Suite input sanitization check

Types of Cross-site scripting

- ▶ Reflected XSS
- ▶ Stored XSS
- ▶ DOM Based XSS
- ▶ Blind XSS

Reflected XSS

- ▶ <https://tryhackme.com/room/xss>

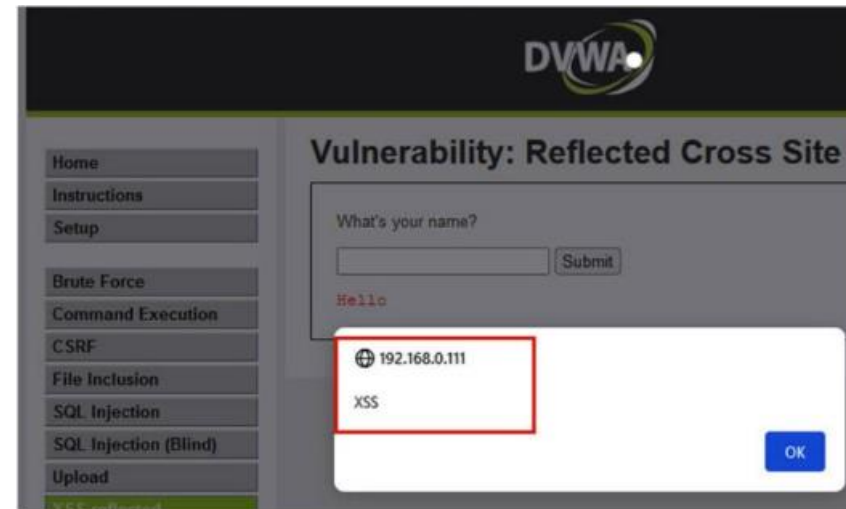
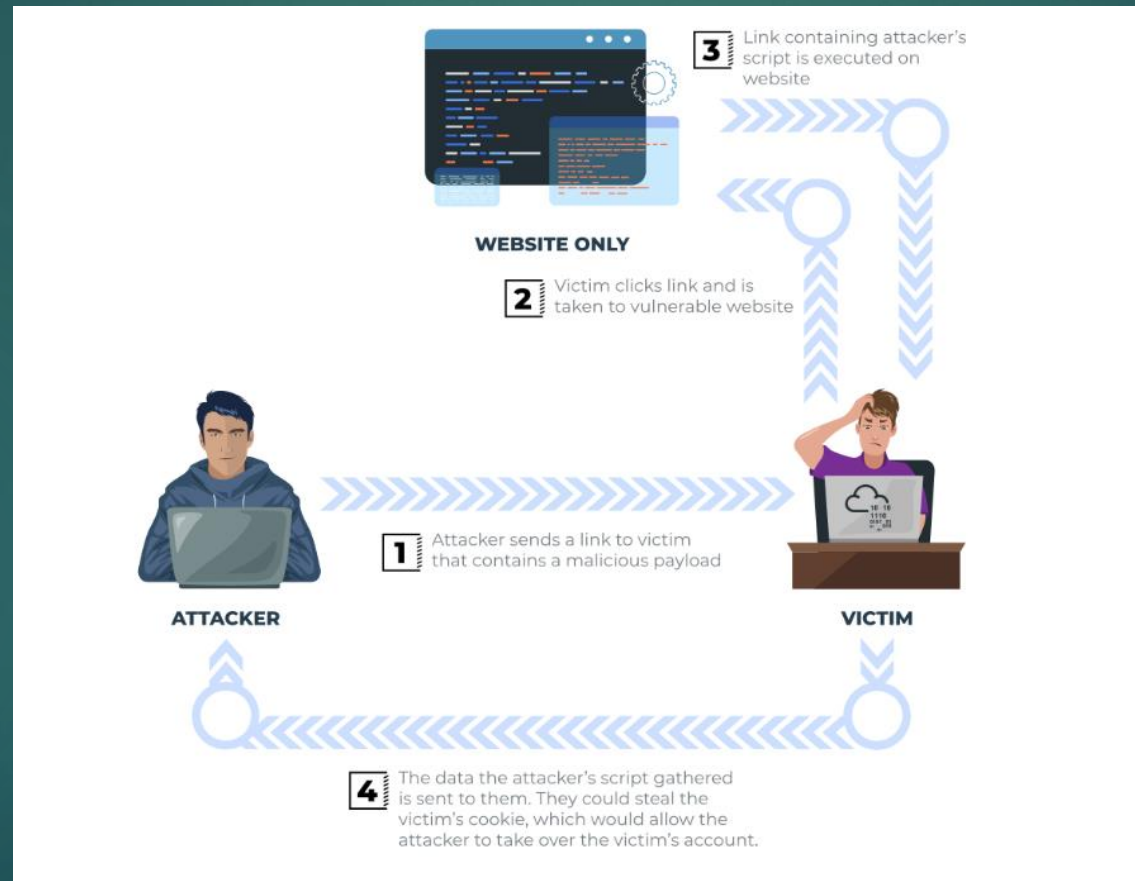


Figure 4.20: DVWA XSS

Reflected XSS



Stored XSS

- ▶ XSS Payload is stored on the web application (database for example)
- ▶ Payload runs when other user visit the site

16



DOM Based XSS

- ▶ DOM
- ▶ Exploitation du DOM
- ▶ Example Scenario
- ▶ Des liens manipulés pourraient être envoyés à des victimes potentielles, les redirigeant vers un autre site web ou volant du contenu de la page ou de la session de l'utilisateur.

Exploitation:

- Lire toutes les données que l'utilisateur peut accéder
- Injecter une fonctionnalité de cheval de Troie dans le site web
- Effectuer un défacement virtuel du site web
- Effectuer toute action que l'utilisateur peut effectuer
- Capturer les identifiants de connexion de l'utilisateur

DOM Based XSS

- ▶ Encodage des données en sortie
- ▶ Filtrage des données d'entrée
- ▶ Utilisation d'en-têtes de réponse appropriée :
- ▶ Définition du flag httponly pour les cookies
- ▶ Politique de sécurité du contenu

Server-Side Request Forgery (SSRF)

- ▶ Exploite des fonctionnalités sur le serveur pour lire ou modifier des ressources internes.
- ▶ Permet à l'attaquant de contraindre l'application côté serveur à effectuer des requêtes HTTP vers un domaine spécifique choisi par l'attaquant.

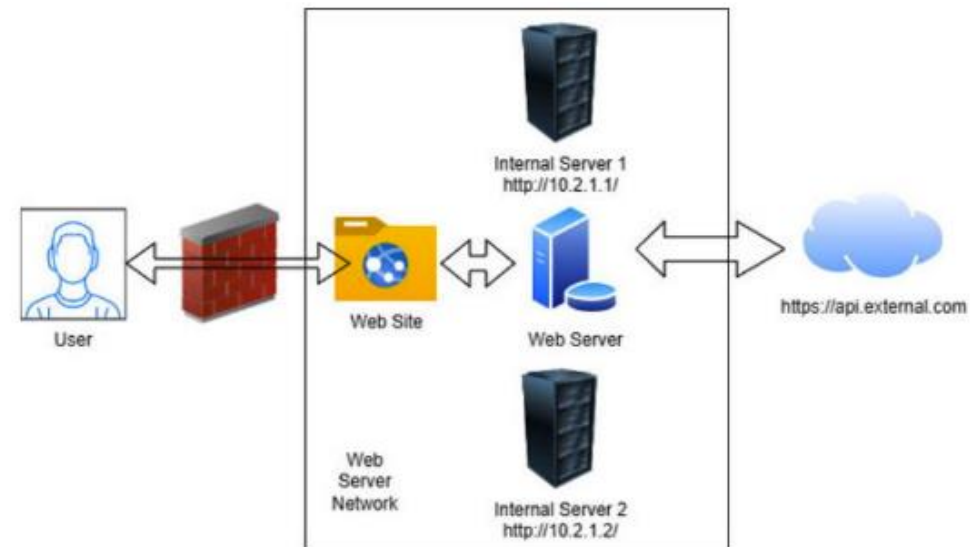


Figure 4.33: SSRF example

SSRF

Exploitations

- ▶ Accès à des zones non autorisé
- ▶ Accès aux données du client/organisation
- ▶ Capacité de s'étendre aux réseaux internes.

Mesures de protection possibles

- ▶ Principe de défense en profondeur
- ▶ Validation des entrées
- ▶ Validation des adresses IP et des noms de domaine

SSRF

Where to look

- Quand un URL entier est utilisé dans un paramètre dans la barre d'adresse
- URL partielle comme un nom d'hôte
- Chemin d'accès de l'URL.

When a full URL is used in a parameter in the address bar:



A hidden field in a form:

```
9 <form method="post" action="/form">
10 <input type="hidden" name="server" value="http://server.website.thm/store">
11 <div>Your Name:</div>
12 <div><input name="client_name"></div>
13 <div>Your Email:</div>
14 <div><input name="client_email"></div>
15 <div>Your Message:</div>
16 <div><textarea name="client_message"></textarea></div>
17 </form>
```

A partial URL such as just the hostname:



Or perhaps only the path of the URL:



Qu'est-ce qu'un IDOR

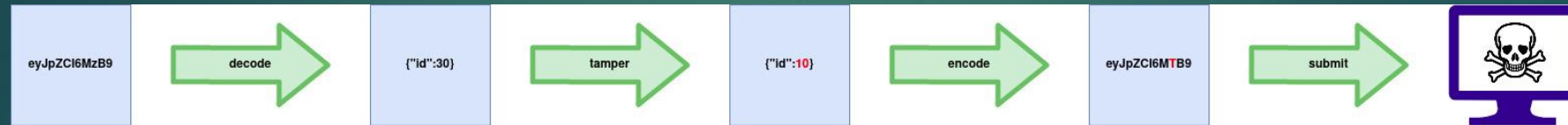
- ▶ Signifie «Insecure Direct Object Reference».
- ▶ Type de vulnérabilité de contrôle d'accès
- ▶ Survient lorsqu'un serveur web reçoit des données fournies par l'utilisateur pour récupérer des objets.
- ▶ Confiance excessive, données d'entrées ne sont pas validées pour confirmer que l'objet appartient à l'utilisateur le demandant

Exemple

http://online-service.thm/profile?user_id=1305 → http://online-service.thm/profile?user_id=1000

IDOR

Finding IDOR's in Encoded Ids



Finding IDOR's in Hashed Ids

Finding IDOR's in Unpredictable Ids

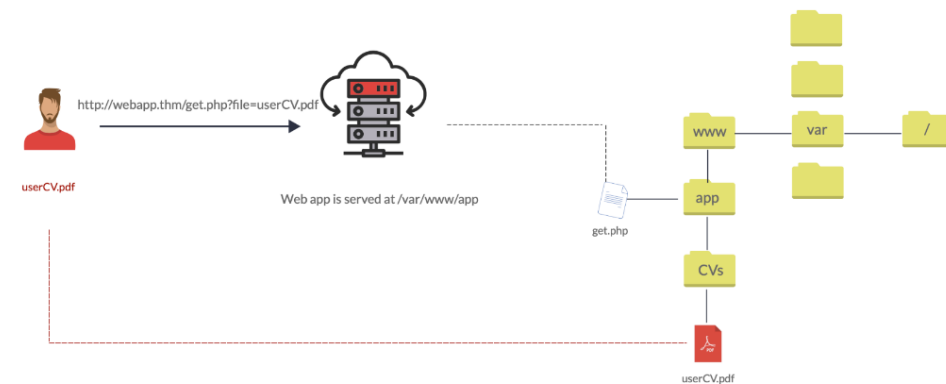
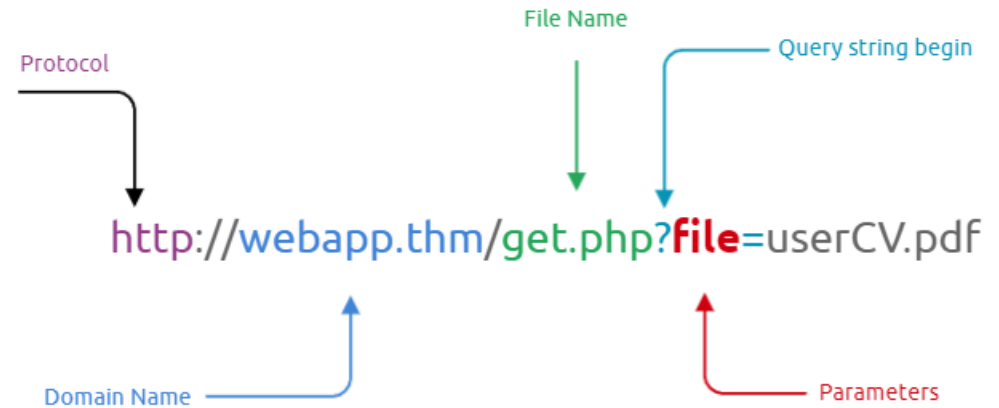
File Inclusion

What is File inclusion?

- ▶ Pratique qui permet à une application de charger et utiliser des fichiers externes, tels que des scripts dans ses opérations

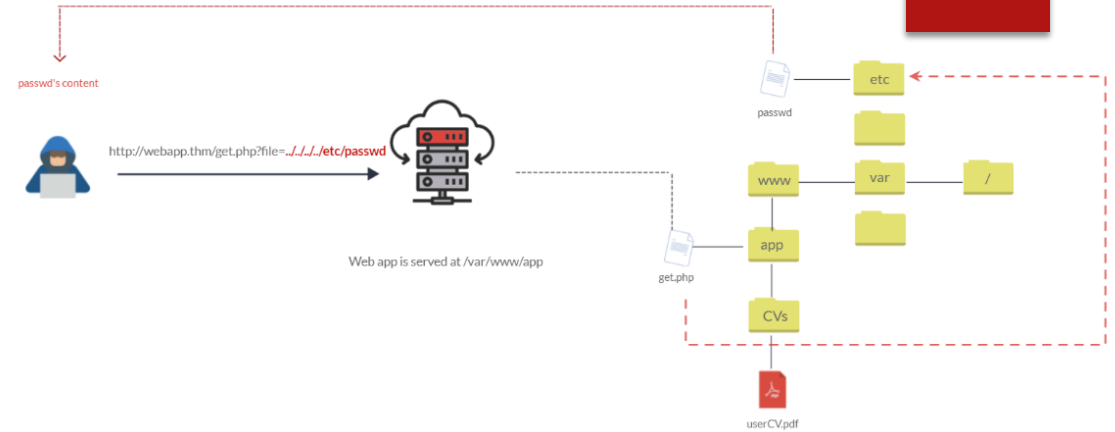
Risques?

- ▶ Accès à des fichiers sensibles, exécuter du code malveillant ou compromettre la sécurité de l'application



File inclusion

Path traversal (directory transversal)



As a result, the web application sends back the file's content to the user.



Local file inclusion (LFI)

- ▶ Vulnérabilités LFI souvent liées au manque de sensibilisation à la sécurité chez les développeurs
- ▶ functions like `include`, `require`, etc., in PHP or similar functions in other languages.
- ▶ LFI vulnerabilities occur in various languages like ASP, JSP, Node.js, and in this case, PHP is used for demonstration.
- ▶ When the directory is specified in the include function, such as `include("languages/" . $_GET['lang'])` → <http://webapp.thm/index.php?lang=../../../../etc/passwd>.
- ▶ LFI exploits leverage similar concepts to path traversal, using the include function's flexibility to load external files into the current page context.

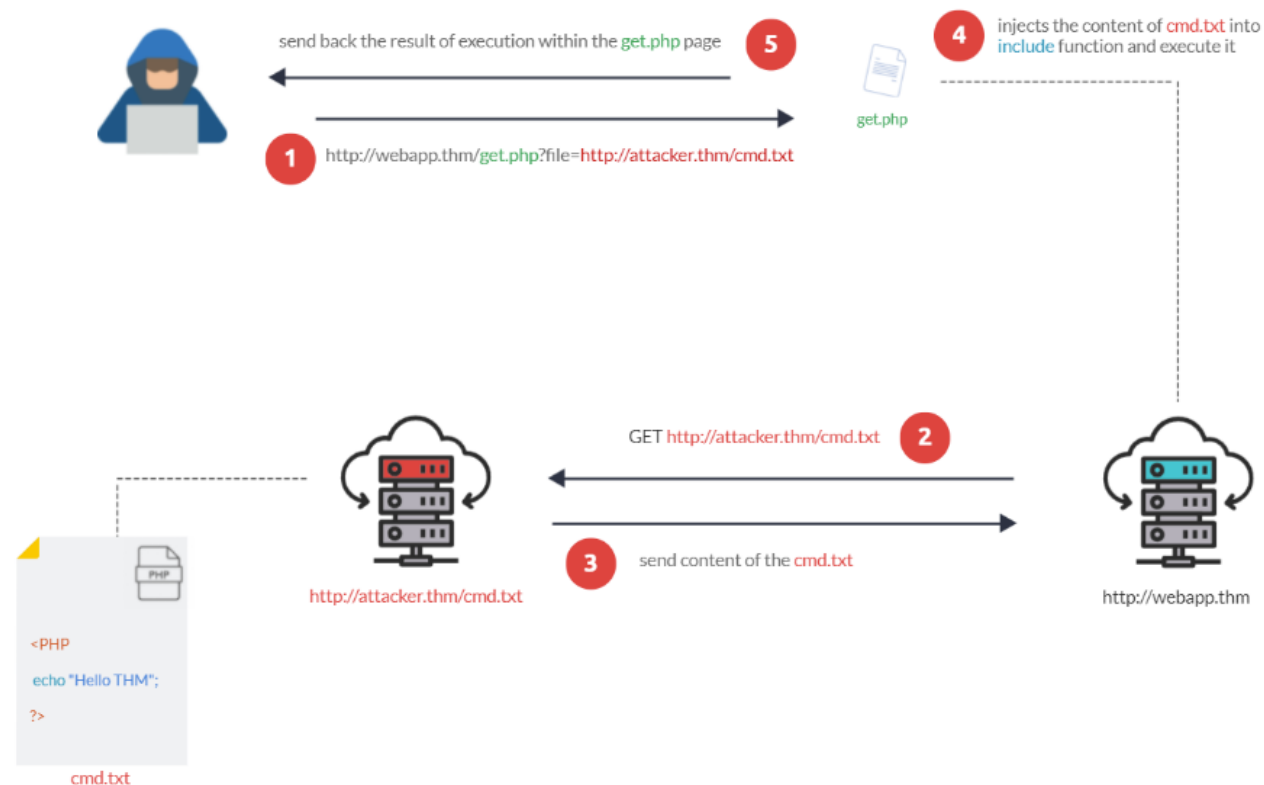
Remote File inclusion

- ▶ Remote File Inclusion (RFI) involves including remote files into a vulnerable application by exploiting input sanitization issues, similar to Local File Inclusion (LFI).
- ▶ RFI requires the 'allow_url_fopen' option to be enabled, allowing inclusion of remote files via URLs.
- ▶ RFI poses a higher risk than LFI as it can lead to Remote Command Execution (RCE) on the server.
- ▶ Consequences of successful RFI attacks include Sensitive Information Disclosure, Cross-site Scripting (XSS), and Denial of Service (DoS).
- ▶ Attackers host malicious files on their server and inject them into the vulnerable application server through the include function using HTTP requests.
- ▶ The content of the injected malicious file executes within the context of the vulnerable application server, allowing the attacker to exploit the system.

RFI - Steps

Remediation

- ▶ Latest version of system and services
- ▶ Turn off PHP errors
- ▶ Web Application Firewall
- ▶ Disable some PHP feature (allow url open, allow url include)
- ▶ Never trust user input → validation against file inclusion
- ▶ White listing for file name and locations as well as blacklisting



Command Injection

What is it ?

- ▶ Web vulnerability → exploit application's behavior to execute commands on the operating system
- ▶ Attacker can run commands using the privileges of the application
- ▶ Allows remote interaction with vulnerable systems

Understanding involves

- ▶ Recognizing its impact and risks on applications
- ▶ Learning to discover and test for this vulnerability
- ▶ Different operating system's payloads
- ▶ Implementing preventive measures to secure applications

Most common API security problems

30

Sample Footer Text
1/11/2024

- ▶ Return data in JSON or XML
- ▶ Rest API
- ▶ Soap API
- ▶ GraphQL API
- ▶ API-Centric Applications

SOAP API

▶ Request:

DELETE / HTTPS/1.1

Host: example.s3.amazonaws.com

```
<DeleteBucket xmlns="http://doc.s3.amazonaws.com/2006-03-01">  
<Bucket>quotes</Bucket> <AWSAccessKeyId>  
AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId> <Timestamp>2006-03-  
01T12:00:00.183Z</Timestamp>  
<Signature>luyz3d3P0aTou39dzbqaEXAMPLE=</Signature>  
</DeleteBucket>
```

➤ Response:

```
<DeleteBucketResponse xmlns="http://s3.amazonaws.com/doc/2006-03-  
01"> <DeleteBucketResponse> <Code>204</Code> <Description>No  
Content</Description> </DeleteBucketResponse>  
</DeleteBucketResponse>
```

GraphQL API

▶ Request:

```
query { shop { name primaryDomain { url host } } }
```

➤ Response:

```
{ "data": { "shop": { "name": "example", "primaryDomain": { "url":  
"https://example.myshopify.com", "host": "example.myshopify.com" } } }  
}
```

API-Centric Applications

- ▶ Composent côté client qui requête et rends des données du server en utilisant des appels API.
- ▶ Bcp d'applications mobile sont construites de cette façon : But sauver du temps car pas besoin de réimplémenter les mêmes fonctionnalités.

Vulnérabilités et sécurité

- ▶ Broken Access Control and Info Leaks
- ▶ Technical Bugs

Broken Access Control and Info leaks

35

Sample Footer Text
1/11/2024

Broken Access Control

- ▶ Manque d'autorisation
- ▶ Défauts de conception
- ▶ Impacts potentiels

Information Leakage

- ▶ Messages d'erreur inappropriés
- ▶ Données sensibles
- ▶ Conséquences

Testing for Technical Bugs

36

- ▶ Server-Side Request Forgery (SSRF) :
API peut accéder à des URL externes ou interne , permettant à un attaquant de déclencher des requêtes vers des ressources arbitraires, ou internes.
- ▶ Path Traversal
Permet à un attaquant de naviguer au-delà du répertoire prévu
- ▶ Insecure File Inclusion
Exécution de code malveillant en insérant des fichiers externes dans les requêtes
- ▶ Insecure Deserialization
Modifier le flux de données ou d'exécuter du code malveillant lorsque désérialisation des données n'est pas sécurisé
- ▶ XML External Entity (XXE)
Risque: Exploitation d'un fichier XML analyseur pour accéder à des ressources externes.
- ▶ Cross-Site Scripting (XSS)
Impact: Failles XSS permettent d'injecter du code malveillant dans pages web consultées par d'autres utilisateurs

Bibliographie

- ▶ Tryhackme : Jr Penetration Tester
<https://tryhackme.com/>
- ▶ Bug Bounty Bootcamp : Vickie Li
- ▶ **Penetration Testing for Jobseekers**
<https://univ.scholarvox.com/catalog/book/docid/88937955?searchterm=Penetration%20Testing%20for%20Jobseekers>