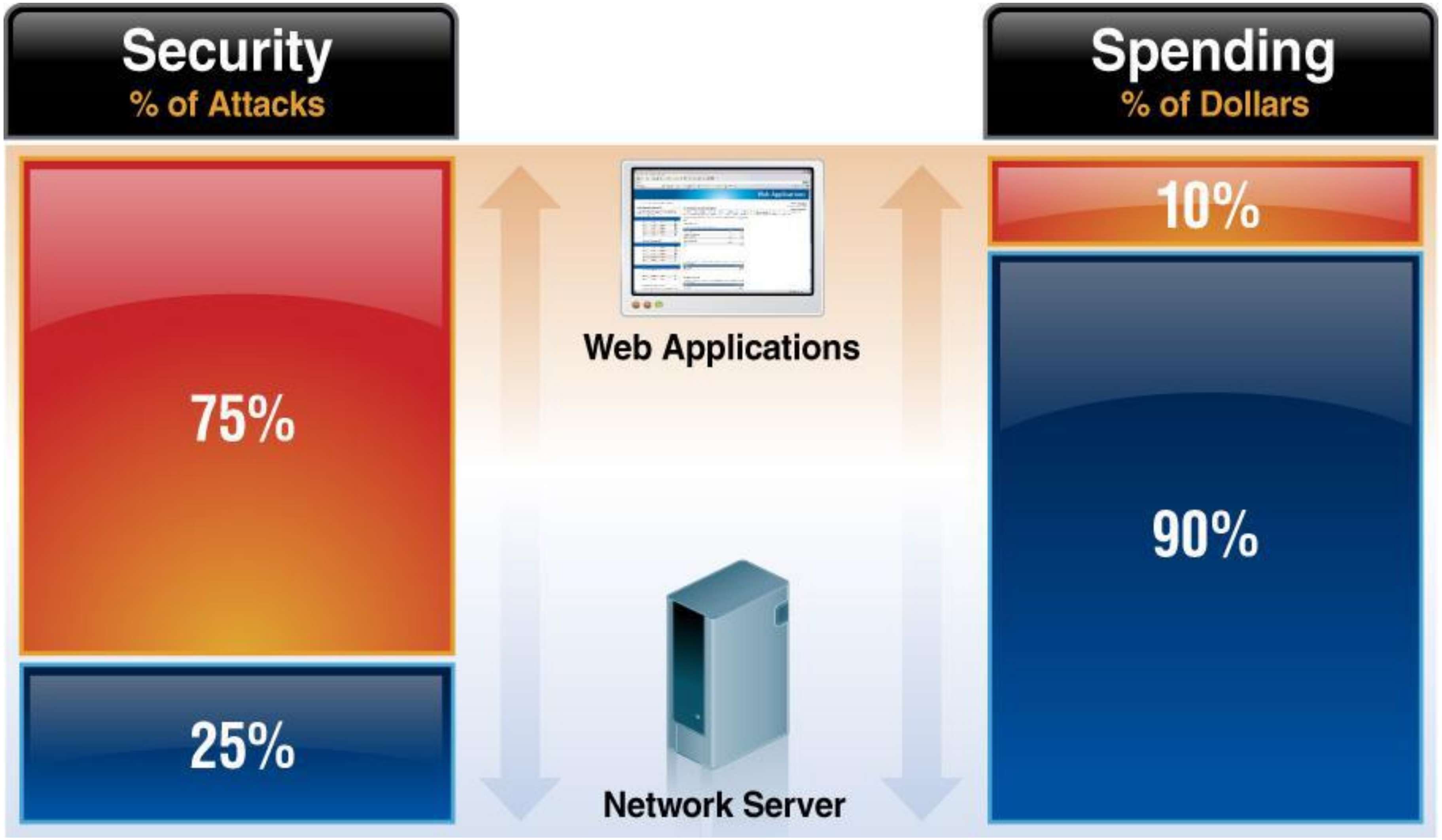
تست امنیتی پایگاه داده و برنامه‌های تحت وب

وحید زنگنه

v.zangeneh@aut.ac.ir

اهمیت امنیت برنامه‌های تحت وب

- Gartner



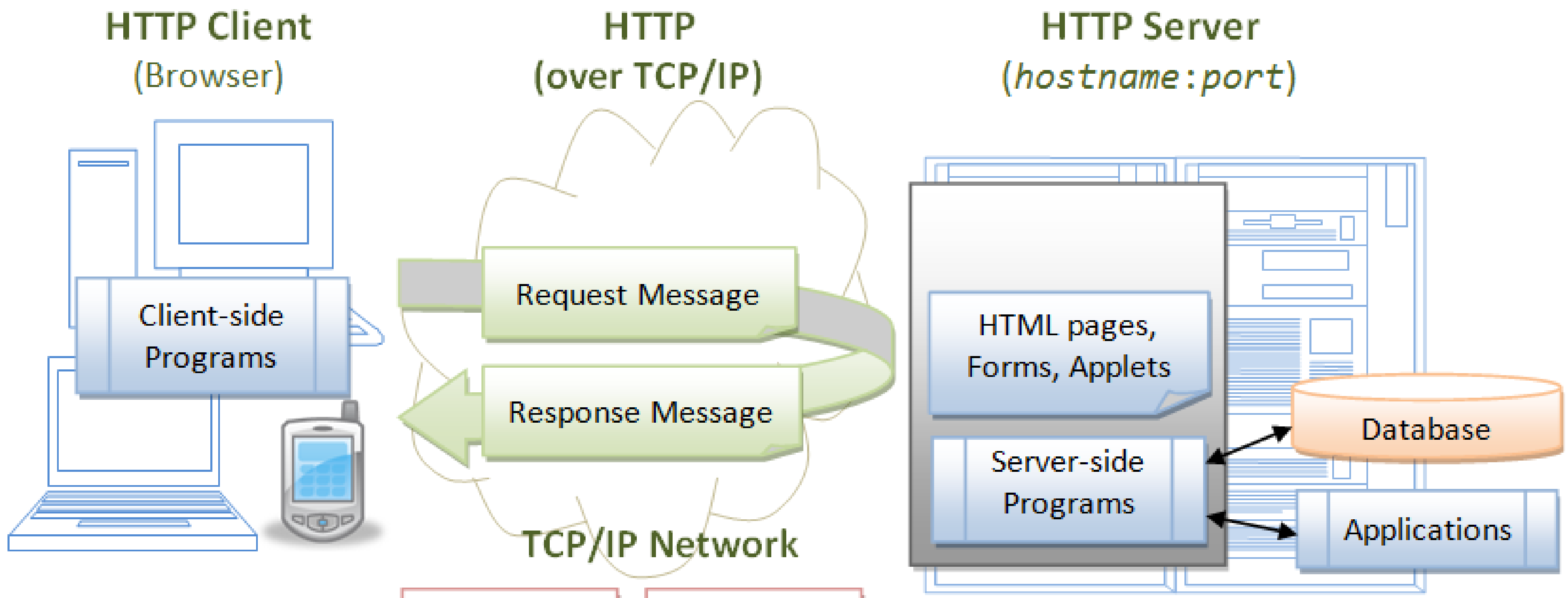
اهمیت امنیت برنامه‌های تحت وب

- ۷۵ درصد حملات در لایه برنامه کاربردی انجام میشود.
- XSS و SQL injection در رده های اول و دوم آسیب پذیری های گزارش شده است.
- ۹۰ درصد سایت ها به حملات بر پایه برنامه های کاربردی هستند. Watchfire
- ۷۸ درصد آسیب پذیری هایی که به آسانی قبل اکسپلویت هستند در برنامه های کاربردی وب قرار دارند. Symantec
- ۸۰ درصد ارگان ها، حوادثی مرتبط با امنیت برنامه های کاربردی را تا سال ۲۰۱۳ تجربه خواهند کرد. Gartner

برنامه های وب هدف ارزشمندی برای نفوذگران می‌باشند

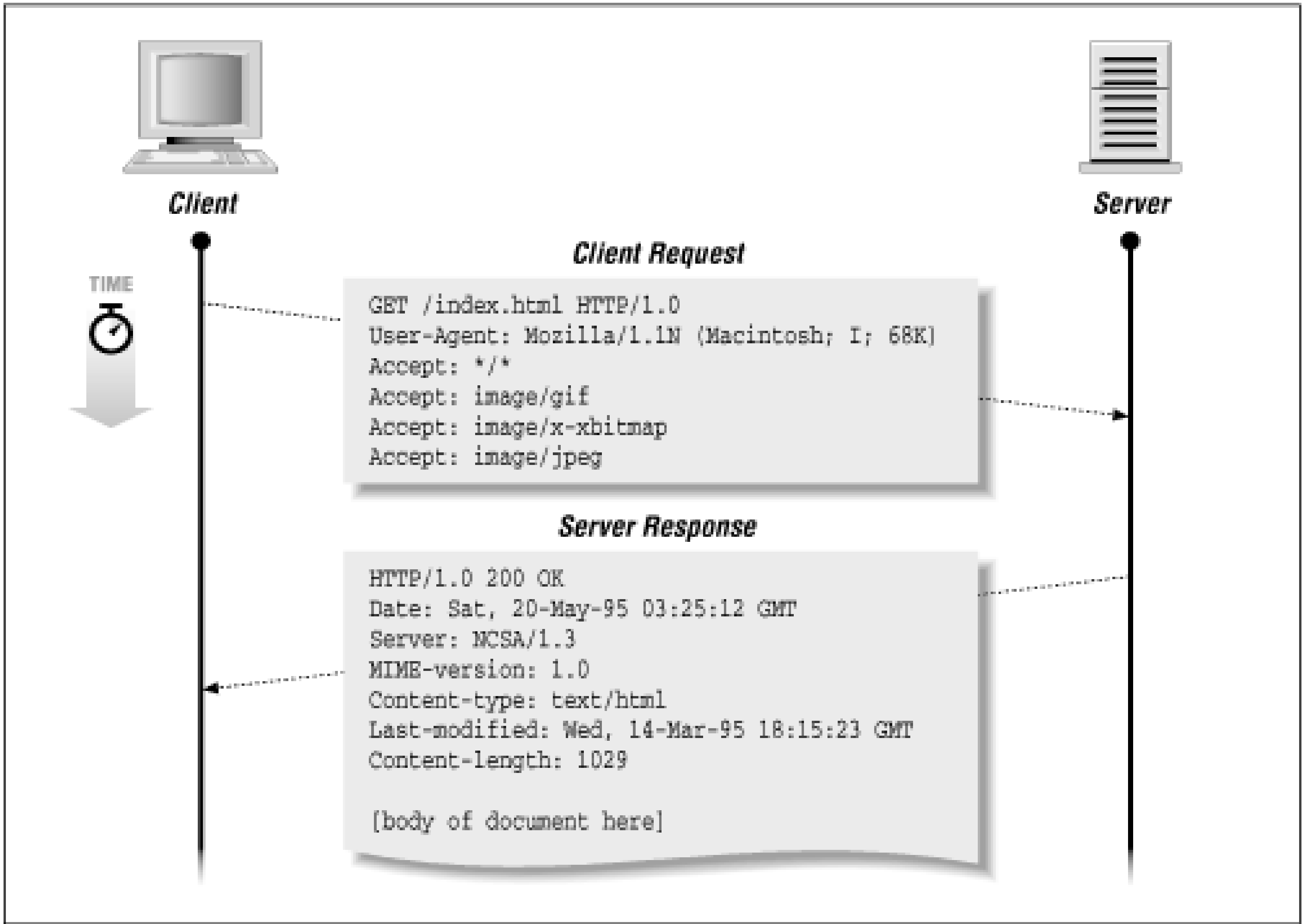
- برنامه های وب به خودی خود ارزش چندانی ندارند (بیشتر به عنوان GUI شناخته می شوند)
- تمام طلاعات از برنامه های وب رد می شود.
- برنامه های وب به پایگاه داده مرتبط و متصل هستند
- مهمترین و شناخته ترین سرویس یک سازمان خدمات وب و وب سایت آن است.

آشنایی با چند واژه



Application	**HTTP**	
Presentation	SSL	
Session		
Transport	**TCP**	Multiplexing (Port), Re-transmission
Network	**IP**	Addressing (IP Address), Routing
Data Link	IEEE 802.11x	
Physical		

آشنایی با چند واژه



```
GET /index.html HTTP/1.0
User-Agent: Mozilla/1.1N (Macintosh; I; 68K)
Accept: */*
Accept: image/gif
Accept: image/x-xbitmap
Accept: image/jpeg
```

```
HTTP/1.0 200 OK
Date: Sat, 20-May-95 03:25:12 GMT
Server: NCSA/1.3
MIME-version: 1.0
Content-type: text/html
Last-modified: Wed, 14-Mar-95 18:15:23 GMT
Content-length: 1029

[body of document here]
```

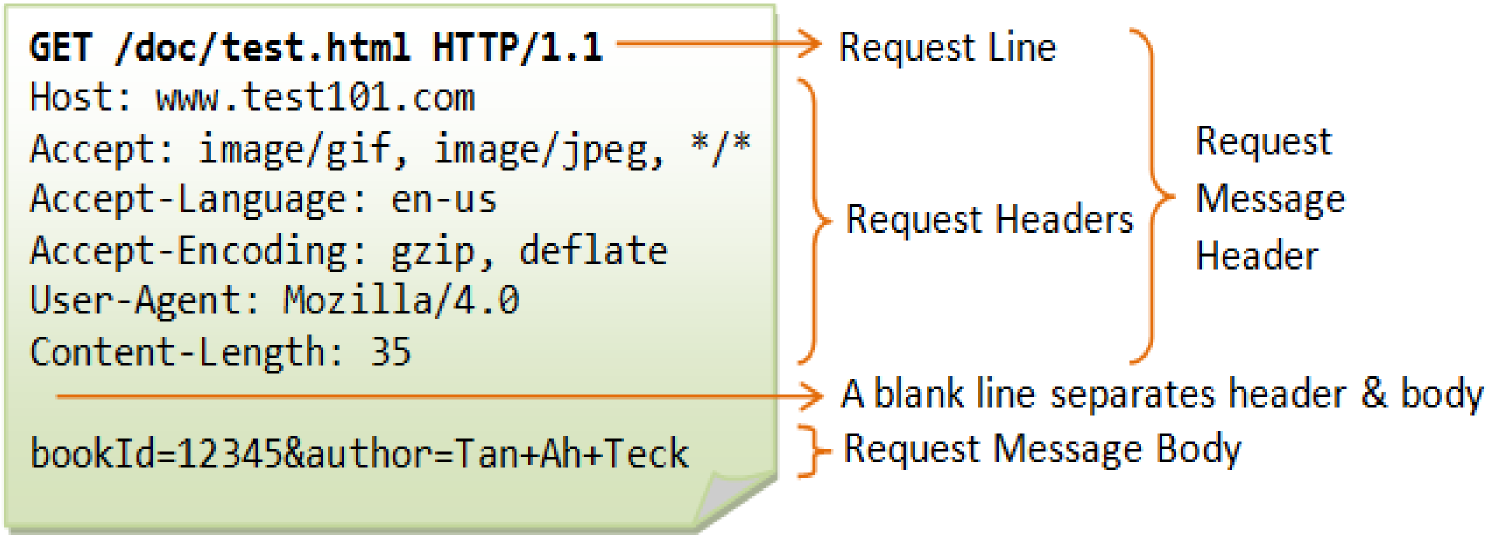
اجزای پیام‌های HTTP

- هر پیام (چه درخواست و چه پاسخ) از سه بخش تشکیل شده است:
- خط درخواست یا خط جواب Request/Response line
- سرآیند ها Headers
- متن پیام Body

قالب درخواست HTTP

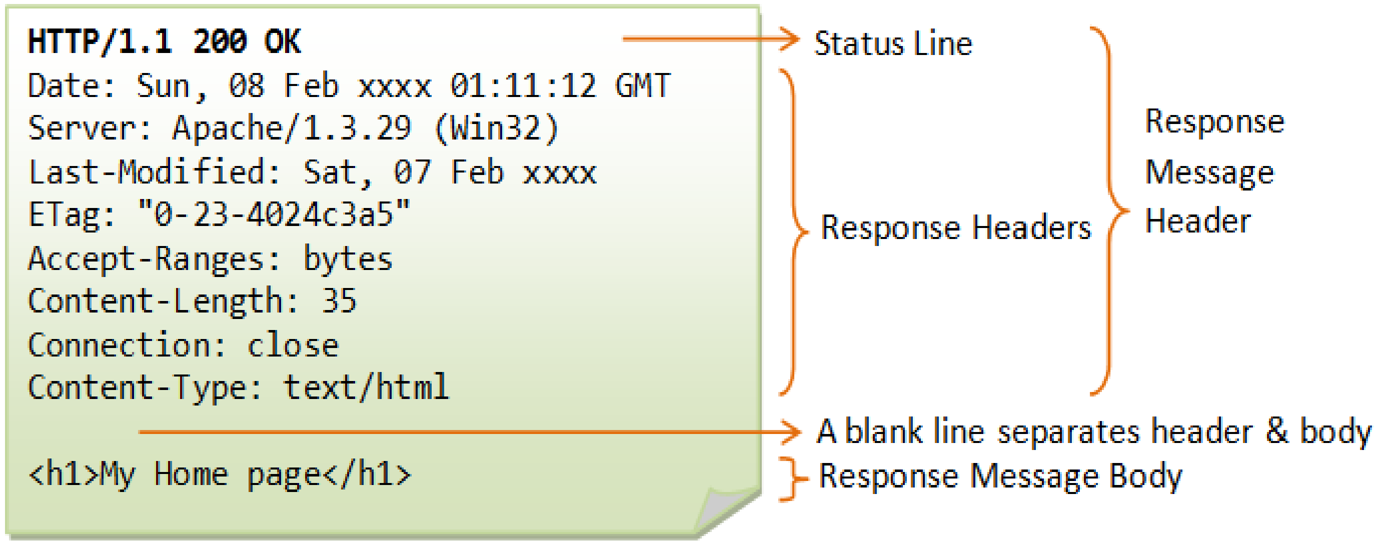
- خط درخواست حاوی چندین بخش است:
- HTTP Method برای مثال GET و POST
- آدرس سند
- نگارش HTTP

- سرآیندهای گوناگونی می توان به درخواست اضافه کرد که مهم ترین این سرآیندها عبارتند از :
  - Cooki
  - Host
  - User-Agent
  - Referrer



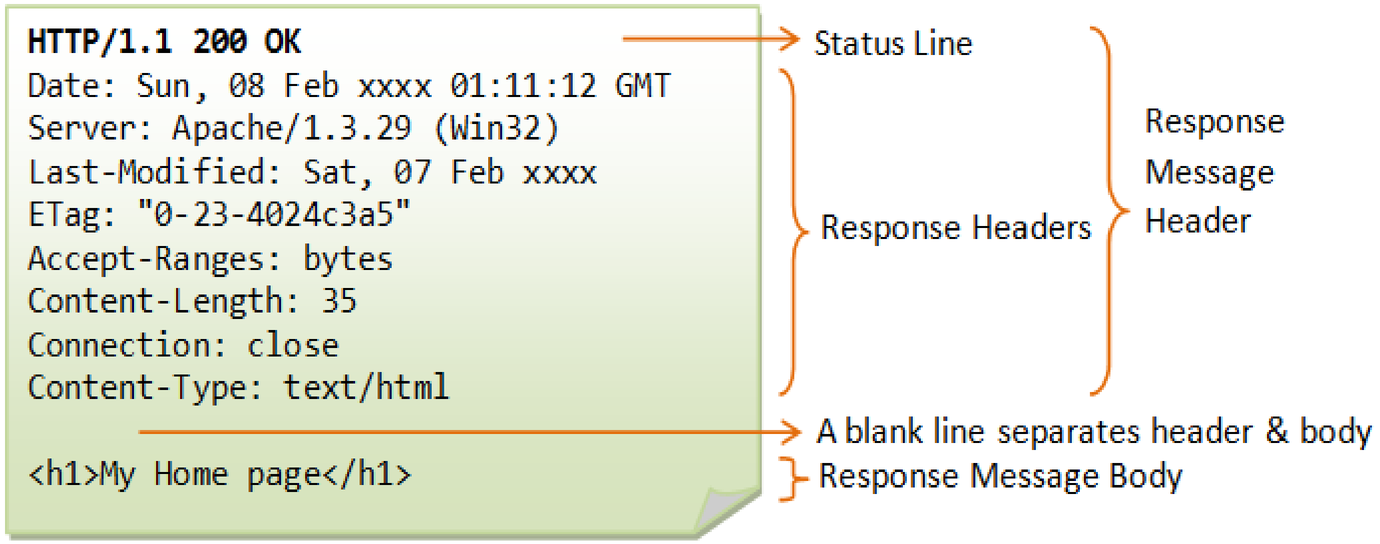
```
GET /doc/test.html HTTP/1.1
Host: www.test101.com
Accept: image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Content-Length: 35

bookId=12345&author=Tan+Ah+Teck
```

قالب پاسخ HTTP



```
HTTP/1.1 200 OK
Date: Sun, 08 Feb xxxx 01:11:12 GMT
Server: Apache/1.3.29 (Win32)
Last-Modified: Sat, 07 Feb xxxx
ETag: "0-23-4024c3a5"
Accept-Ranges: bytes
Content-Length: 35
Connection: close
Content-Type: text/html

<h1>My Home page</h1>
```

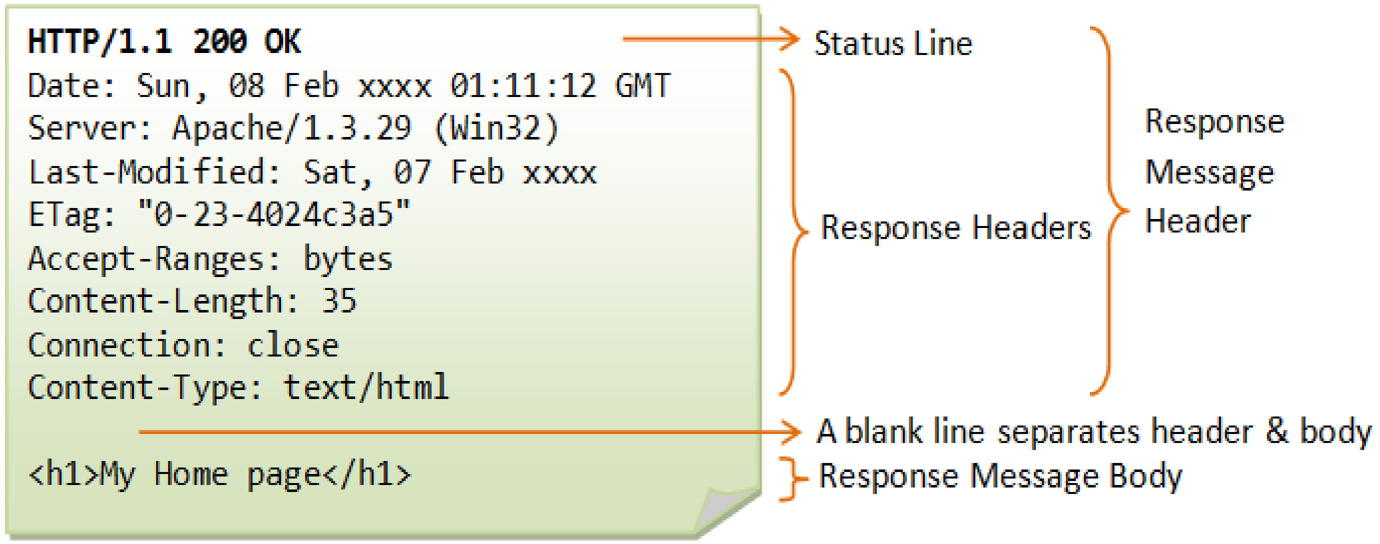
قالب پاسخ HTTP



```
HTTP/1.1 200 OK
Date: Sun, 08 Feb xxxx 01:11:12 GMT
Server: Apache/1.3.29 (Win32)
Last-Modified: Sat, 07 Feb xxxx
ETag: "0-23-4024c3a5"
Accept-Ranges: bytes
Content-Length: 35
Connection: close
Content-Type: text/html

<h1>My Home page</h1>
```

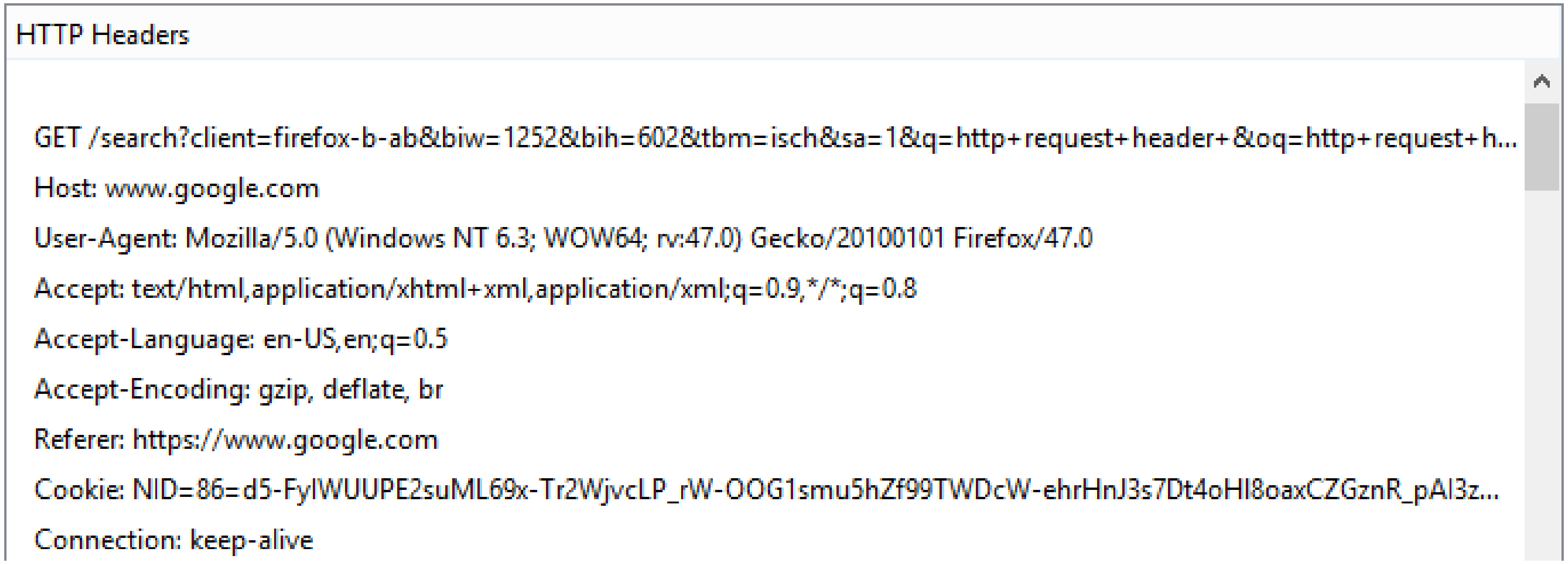
قالب پاسخ HTTP

■



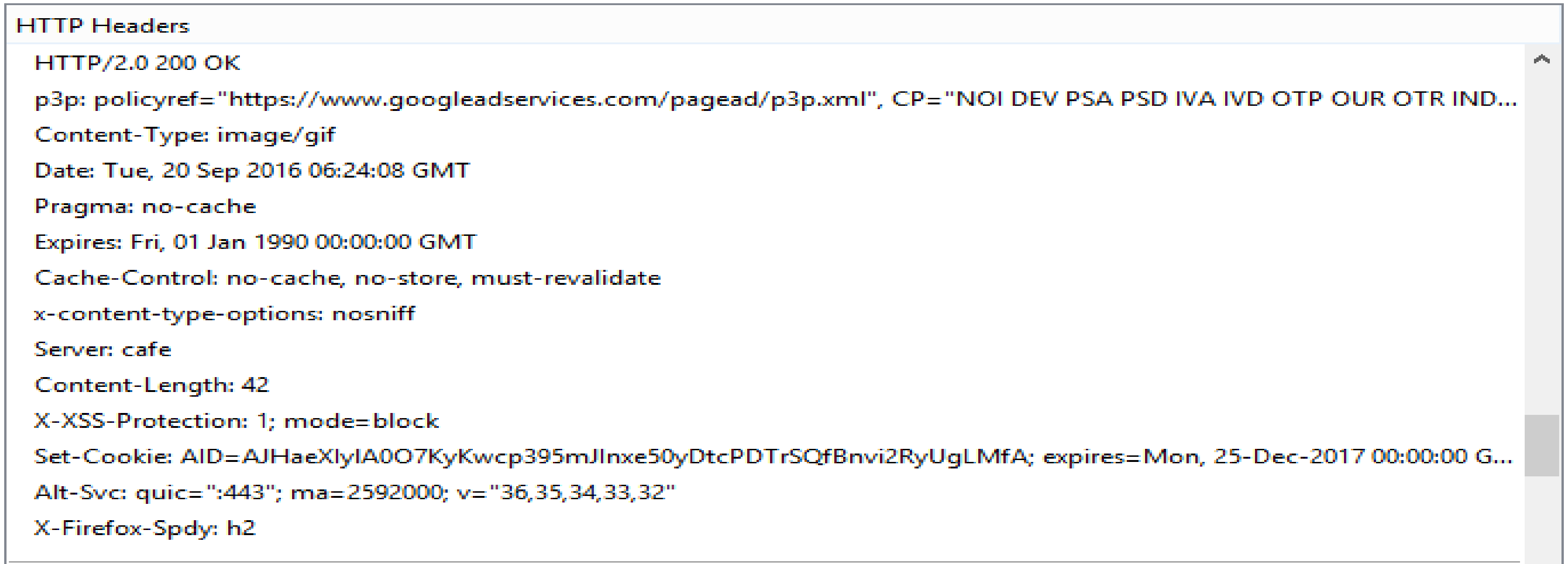
```
HTTP/1.1 200 OK
Date: Sun, 08 Feb xxxx 01:11:12 GMT
Server: Apache/1.3.29 (Win32)
Last-Modified: Sat, 07 Feb xxxx
ETag: "0-23-4024c3a5"
Accept-Ranges: bytes
Content-Length: 35
Connection: close
Content-Type: text/html

<h1>My Home page</h1>
```

نمونه ای از سرآیند درخواست و پاسخ



HTTP Headers

```
GET /search?client=firefox-b-ab&biw=1252&bih=602&tbm=isch&sa=1&q=http+request+header+&oq=http+request+h...
Host: www.google.com
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:47.0) Gecko/20100101 Firefox/47.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://www.google.com
Cookie: NID=86=d5-FyIWUUPE2suML69x-Tr2WjvcLP_rW-OOG1smu5hZf99TWDcW-ehrHnJ3s7Dt4oHl8oaxCZGznR_pAl3z...
Connection: keep-alive
```



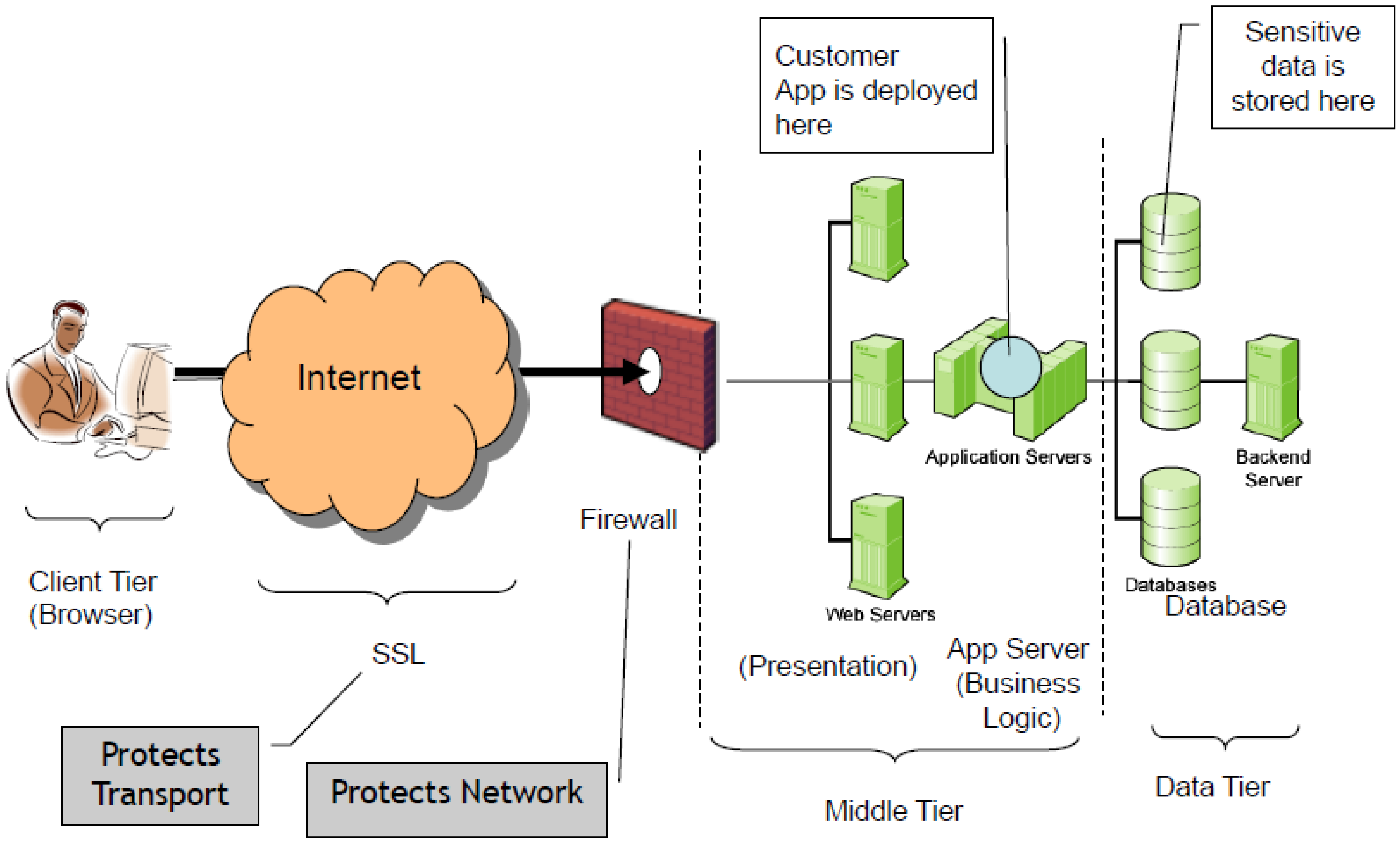
HTTP Headers

```
HTTP/2.0 200 OK
p3p: policyref="https://www.googleadservices.com/pagead/p3p.xml", CP="NOI DEV PSA PSD IVA IVD OTP OUR OTR IND...
Content-Type: image/gif
Date: Tue, 20 Sep 2016 06:24:08 GMT
Pragma: no-cache
Expires: Fri, 01 Jan 1990 00:00:00 GMT
Cache-Control: no-cache, no-store, must-revalidate
x-content-type-options: nosniff
Server: cafe
Content-Length: 42
X-XSS-Protection: 1; mode=block
Set-Cookie: AID=AJHaeXlyIA0O7KyKwcp395mJInxe50yDtcPDTrSQfBnvi2RyUgLMfA; expires=Mon, 25-Dec-2017 00:00:00 G...
Alt-Svc: quic=":443"; ma=2592000; v="36,35,34,33,32"
X-Firefox-Spdy: h2
```

ارسال یک درخواست ساده

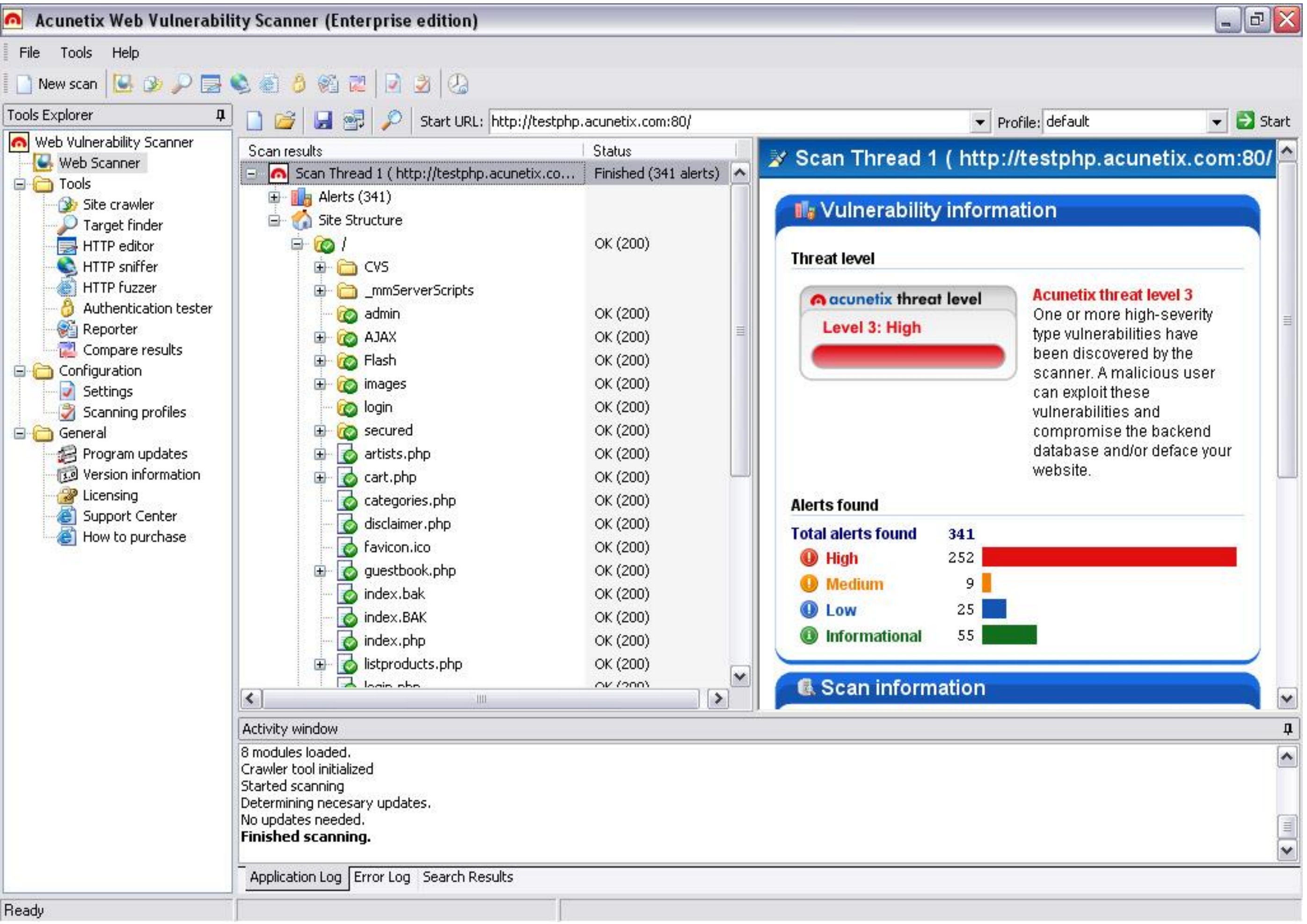
- nc.exe www.google.com 80 <req.txt> res.html
- استفاده از curl برای ارتباط با سرور
- http://curl.haxx.se/download.html
- Curl http://www.google.com

اجزای دخیل در امنیت برنامه‌های وب



آسیب پذیری در سرویس دهنده های وب

- وب سرورها نیز شبیه دیگر برنامه های کاربردی آسیب پذیری های مختلفی دارند نظیر:



Start URL: http://testphp.acunetix.com:80/ Profile: default

Scan results	Status
Scan Thread 1 (http://testphp.acunetix.co...	Finished (341 alerts)
Alerts (341)	
Site Structure	
/	OK (200)
CVS	
_mmServerScripts	
admin	OK (200)
AJAX	OK (200)
Flash	OK (200)
images	OK (200)
login	OK (200)
secured	OK (200)
artists.php	OK (200)
cart.php	OK (200)
categories.php	OK (200)
disclaimer.php	OK (200)
favicon.ico	OK (200)
guestbook.php	OK (200)
index.bak	OK (200)
index.BAK	OK (200)
index.php	OK (200)
listproducts.php	OK (200)

Scan Thread 1 (http://testphp.acunetix.com:80/

Threat level

acunetix threat level
Level 3: High

Acunetix threat level 3
One or more high-severity type vulnerabilities have been discovered by the scanner. A malicious user can exploit these vulnerabilities and compromise the backend database and/or deface your website.

Alerts found

Total alerts found	341
High	252
Medium	9
Low	25
Informational	55

```
8 modules loaded.
Crawler tool initialized
Started scanning
Determining necesary updates.
No updates needed.
Finished scanning.
```

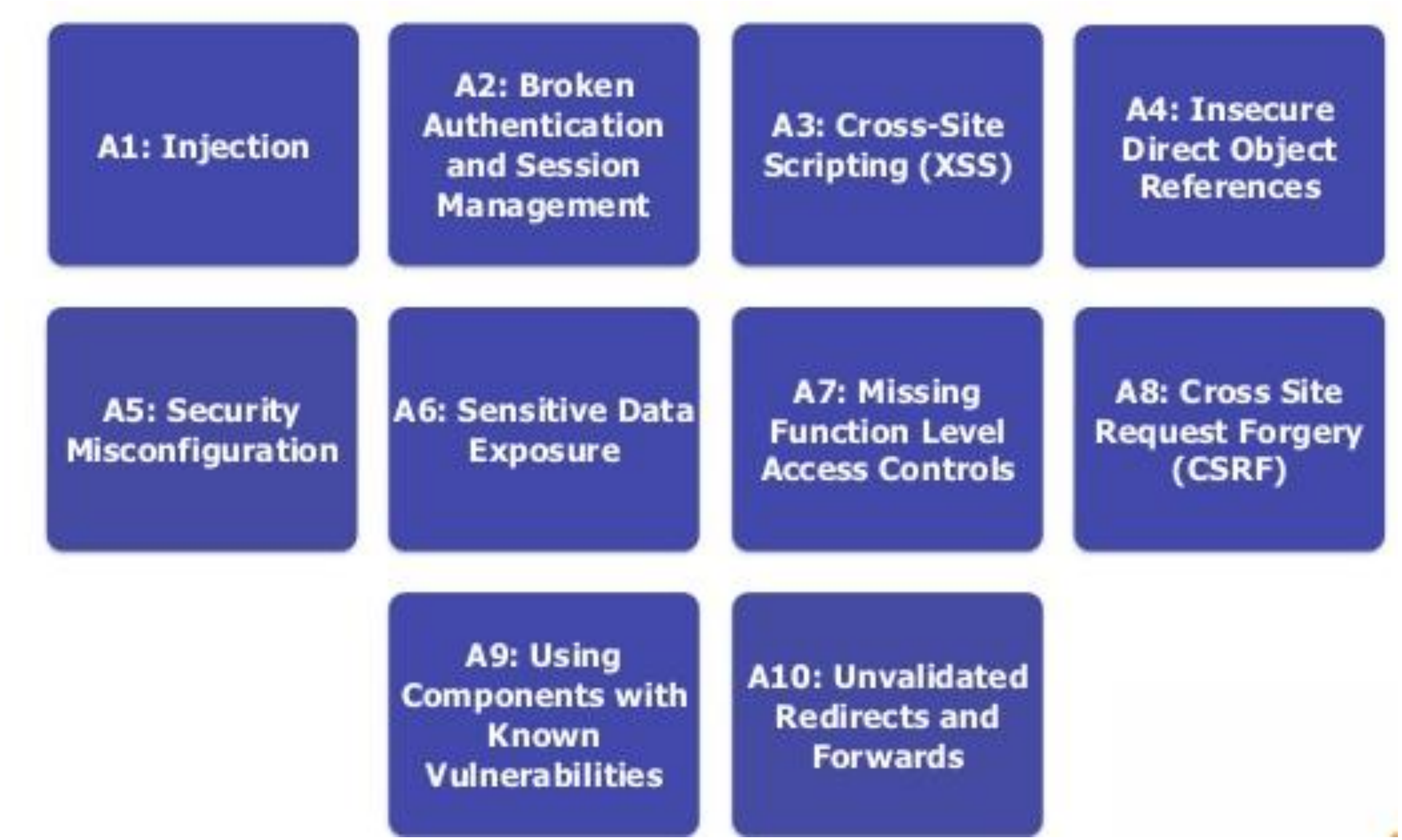
- Buffer overflow
- Format String
- Directory Traversal
- Memory leak
- DOS
- …

- برای کشف:
- Whisker, N-stealth GFI Language, Nessus, ….

آسیب پذیری در نرم افزارهای تحت وب

- OWASP
  - Open Web Application Security Project
- مجموعه ای شرکت ها، موسسات تحقیقاتی و دانشگاهی که در حوزه امنیت نرم افزار کاربردی کار میکنند.
  - مستندات و Guide line
  - استاندارد
  - ابزارهای امنیتی
- https://www.owasp.org

آسیب پذیری ها (بر اساس OWASP TOP 10)



A1: Injection

A2: Broken Authentication and Session Management

A3: Cross-Site Scripting (XSS)

A4: Insecure Direct Object References

A5: Security Misconfiguration

A6: Sensitive Data Exposure

A7: Missing Function Level Access Controls

A8: Cross Site Request Forgery (CSRF)

A9: Using Components with Known Vulnerabilities

A10: Unvalidated Redirects and Forwards

A1. Injection

Injection means…

- Tricking an application into including unintended commands in the data sent to an interpreter

Interpreters…

- Take strings and interpret them as commands
- SQL, OS Shell, LDAP, XPath, Hibernate, etc…

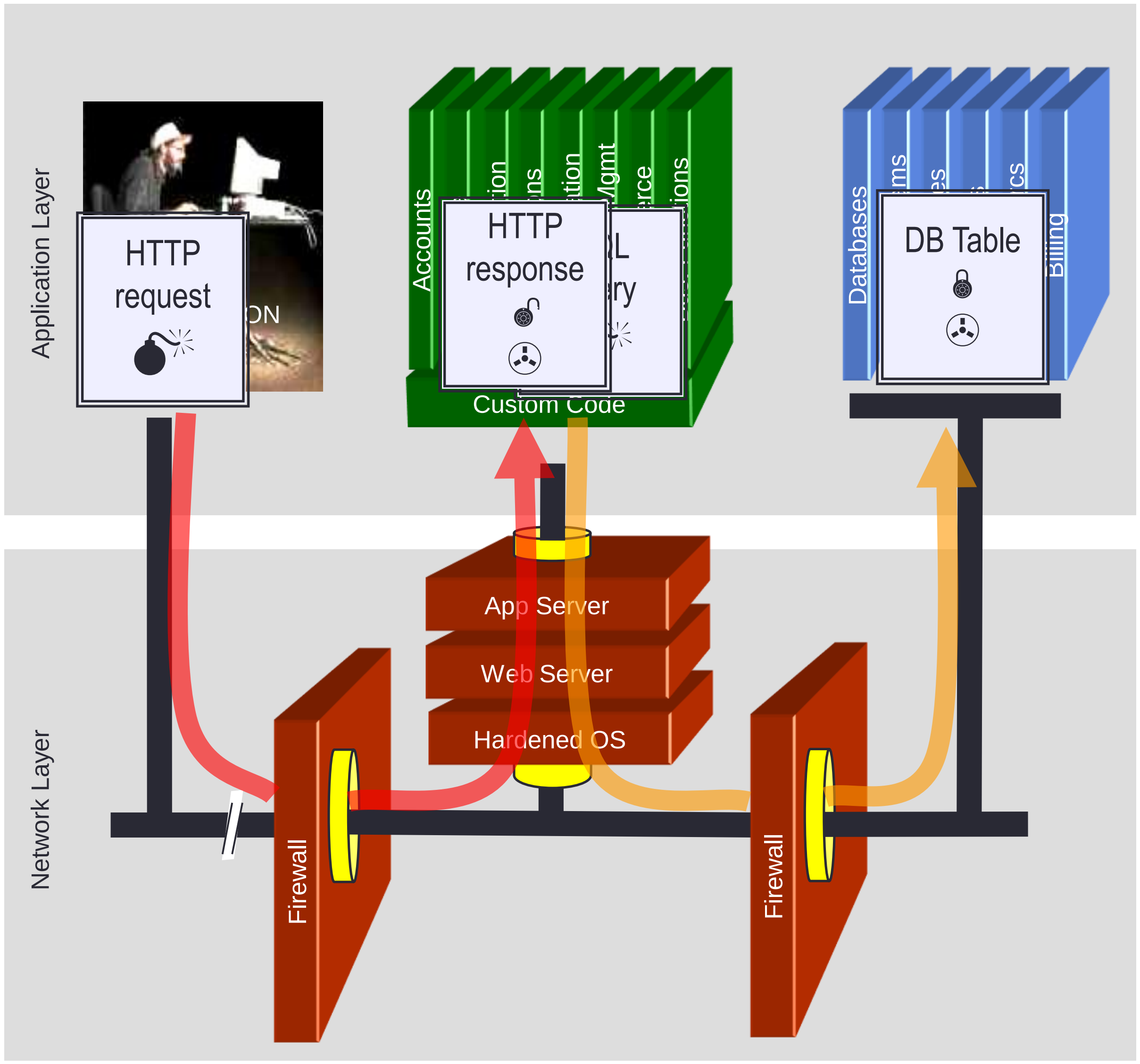
SQL injection is still quite common

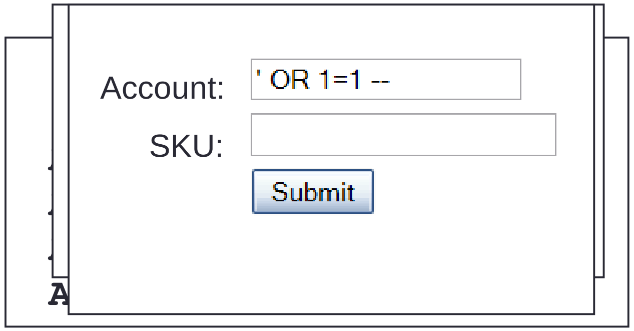
- Many applications still susceptible (really don’t know why)
- Even though it’s usually very simple to avoid

Typical Impact

- Usually severe. Entire database can usually be read or modified
- May also allow full database schema, or account access, or even OS level access

SQL Injection





1. Application presents a form to the attacker
2. Attacker sends an attack in the form data
3. Application forwards attack to the database in a SQL query
4. Database runs query containing attack and sends encrypted results back to application
5. Application decrypts data as normal and sends results to the user

SQL Injection

- جزئیات بیشتر:
- کد برنامه:
- $user=$_POST[“user”];
- $pass=$_POST[“pass”];
- $query=“selsct * from isers where username=‘”.$user.”’ AND password=‘”.$pass.”’”;
- $result=mysql_query($query);

SQL Injection

- ورودی کاربر در فیلد user
- Admin’;--
- ‘OR ‘1’=‘1
- دستور SQL ساخته شده:
- Select * from user where username=‘admin’;-- and password=“;
- Select * from user where username=‘ ‘OR ‘1’=‘1’ and password=“;

SQL Injection

- چرا آسیب پذیری ایجاد می شود؟
- عدم کنترل داده ورودی از سمت کاربر
- چه اثری دارد؟
- اجرای دستور در Engine هدف (SQL DBMS)
- خواندن
- تغییر
- دسترسی به سیستم عامل
- چگونه مورد سوء استفاده قرار میگیرد؟
- با ارسال ورودی حاوی دستورات SQL با فرمت مشخص از سمت مهاجم
- متفاوت بر اساس نوع DBMS
- Access, Mysql, SQL server, Oracel

SQL Injection

- چگونه وجود آسیب پذیری را تشخیص دهیم؟
- ارسال کاراکترهای خاص و تصمیم گیری بر اساس نتیجه
- Errorهای دریافتی
- تاخیر زمانی
- سایر اثرات نشان دهنده اجرا شدن دستور (remote ping)

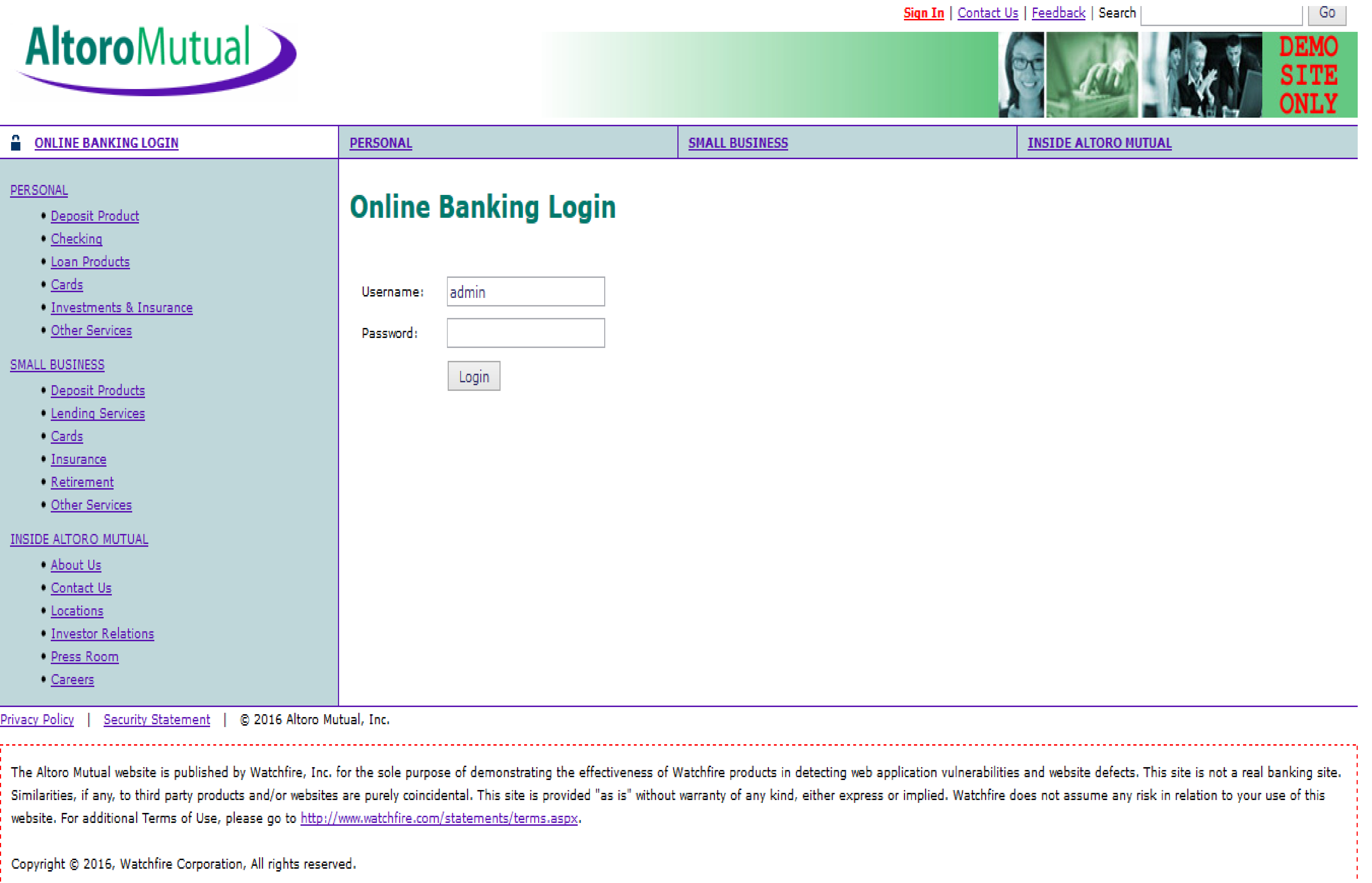
SQL Injection

- چگونه آسیب پذیری را از بین ببریم یا جلوگیری کنیم؟
- اعتبار سنجی داده ورودی
- White-list
- Black-list
- Variable binding
- استفاده از Stored Procedure ها
- دسترسی App به DBMS با کاربر محدود
- عدم نمایش خطاهای بازگشتی

منابع برای مطالعه بیشتر

- SQL Injection Attacks and Defense
- Justin clarke
- SYNGRESS
- OWASP Cheatsheet
- Search for oracle sql injection cheatsheet
- …
- Guide lines from microsoft, oracle, …

نمونه testfire.net/bank/login.aspx



Sign In | Contact Us | Feedback | Search Go

DEMO SITE ONLY

ONLINE BANKING LOGIN | PERSONAL | SMALL BUSINESS | INSIDE ALTORO MUTUAL

PERSONAL
- Deposit Product
- Checking
- Loan Products
- Cards
- Investments & Insurance
- Other Services

SMALL BUSINESS
- Deposit Products
- Lending Services
- Cards
- Insurance
- Retirement
- Other Services

INSIDE ALTORO MUTUAL
- About Us
- Contact Us
- Locations
- Investor Relations
- Press Room
- Careers

Online Banking Login

Username:	admin
Password:	

Login

Privacy Policy | Security Statement | © 2016 Altoro Mutual, Inc.

The Altoro Mutual website is published by Watchfire, Inc. for the sole purpose of demonstrating the effectiveness of Watchfire products in detecting web application vulnerabilities and website defects. This site is not a real banking site. Similarities, if any, to third party products and/or websites are purely coincidental. This site is provided "as is" without warranty of any kind, either express or implied. Watchfire does not assume any risk in relation to your use of this website. For additional Terms of Use, please go to http://www.watchfire.com/statements/terms.aspx.

Copyright © 2016, Watchfire Corporation, All rights reserved.

A2 – Cross-Site Scripting (XSS)

Occurs any time…

- Raw data from attacker is sent to an innocent user’s browser

Raw data…

- Stored in database
- Reflected from web input (form field, hidden field, URL, etc…)
- Sent directly into rich JavaScript client

Virtually <u>every</u> web application has this problem

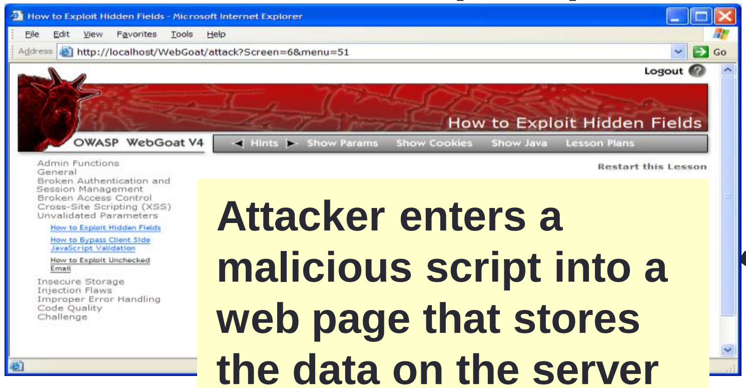
- Try this in your browser – javascript:alert(document.cookie)

Typical Impact

- Steal user’s session, steal sensitive data, rewrite web page, redirect user to phishing or malware site
- Most Severe: Install XSS proxy which allows attacker to observe and direct all user’s behavior on vulnerable site and force user to other sites

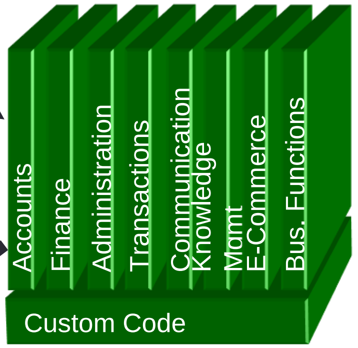
Cross-Site Scripting

1 Attacker sets the trap – update my profile

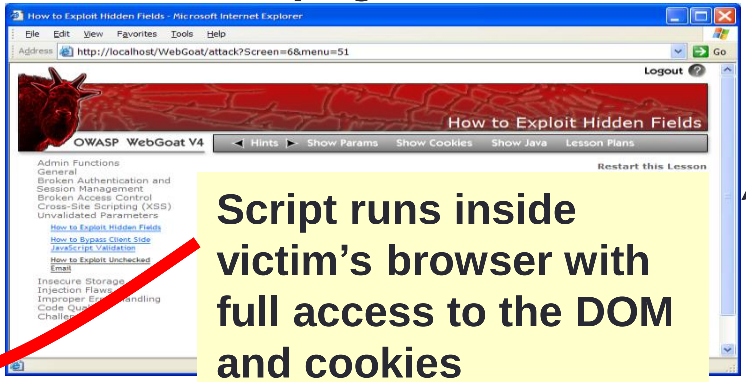

How to Exploit Hidden Fields - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost/WebGoat/attack?Screen=6&menu=51
Logout
How to Exploit Hidden Fields
OWASP WebGoat V4
Hints Show Params Show Cookies Show Java Lesson Plans
Restart this Lesson
Admin Functions
General
Broken Authentication and Session Management
Broken Access Control
Cross-Site Scripting (XSS)
Unvalidated Parameters
How to Exploit Hidden Fields
How to Bypass Client Side JavaScript Validation
How to Exploit Unchecked Email
Insecure Storage
Injection Flaws
Improper Error Handling
Code Quality
Challenge

Attacker enters a malicious script into a web page that stores the data on the server

Application with stored XSS vulnerability



2 Victim views page – sees attacker profile

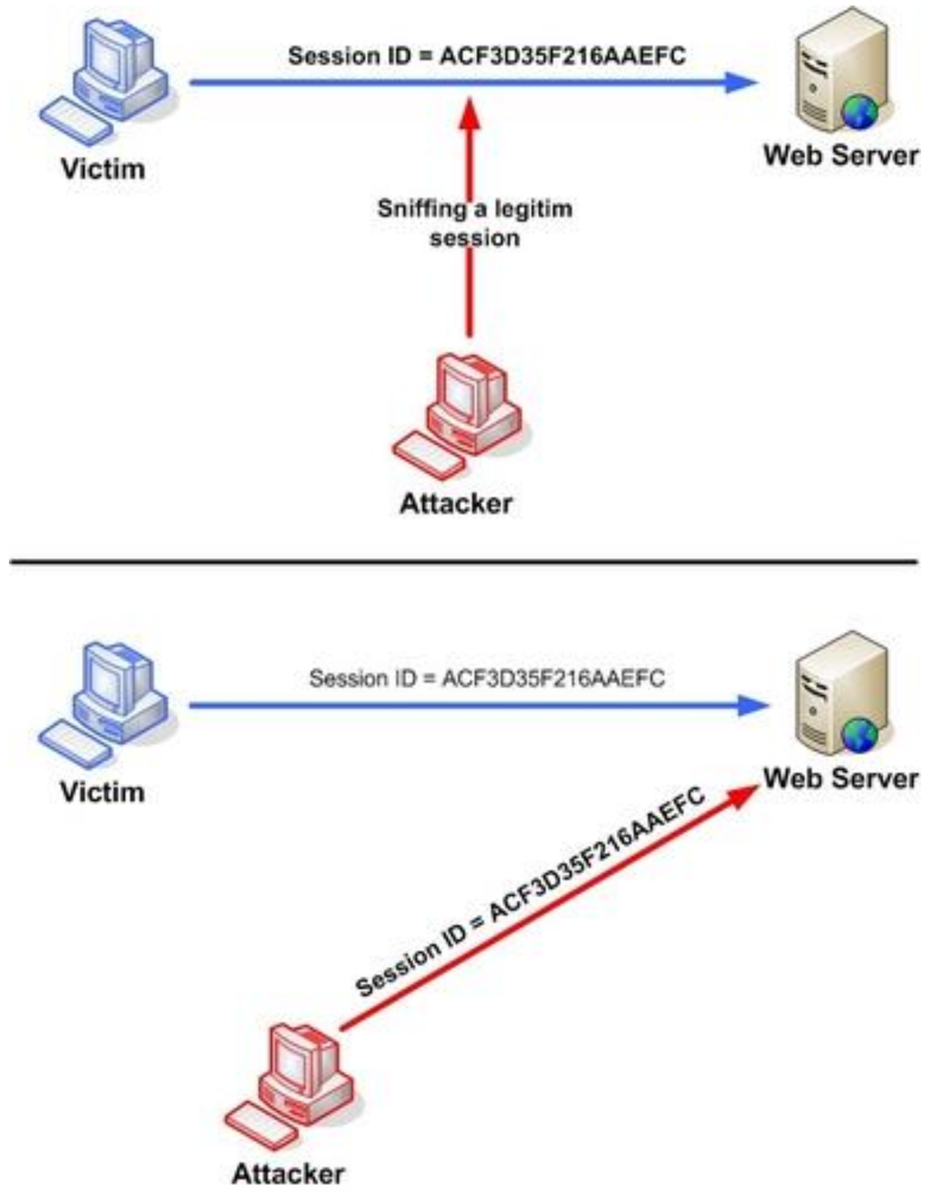

How to Exploit Hidden Fields - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost/WebGoat/attack?Screen=6&menu=51
Logout
How to Exploit Hidden Fields
OWASP WebGoat V4
Hints Show Params Show Cookies Show Java Lesson Plans
Restart this Lesson
Admin Functions
General
Broken Authentication and Session Management
Broken Access Control
Cross-Site Scripting (XSS)
Unvalidated Parameters
How to Exploit Hidden Fields
How to Bypass Client Side JavaScript Validation
How to Exploit Unchecked Email
Insecure Storage
Injection Flaws

Script runs inside victim's browser with full access to the DOM and cookies

3 Script silently sends attacker Victim's session cookie

Cross-Site Scripting

- Session Hijacking
- متود document.cookie
- سایر روشها
- Sniffing


Session ID = ACF3D35F216AAEFC
Victim
Web Server
Sniffing a legitim session
Attacker
Session ID = ACF3D35F216AAEFC
Victim
Web Server
Session ID = ACF3D35F216AAEFC
Attacker

Cross-Site Scripting

- چرا آسیب پذیری ایجاد میشود؟
- عدم کنترل متغییرهای نمایش داده شده در صفحات وب (دلیل اصلی)
- عدم کنترل ورودی
- چه اثری دارد؟
- Session hijacking
- ارسال اطلاعات نامعتبر از طرف کاربر (mail forwarder)
- Redirect به سمت Malware

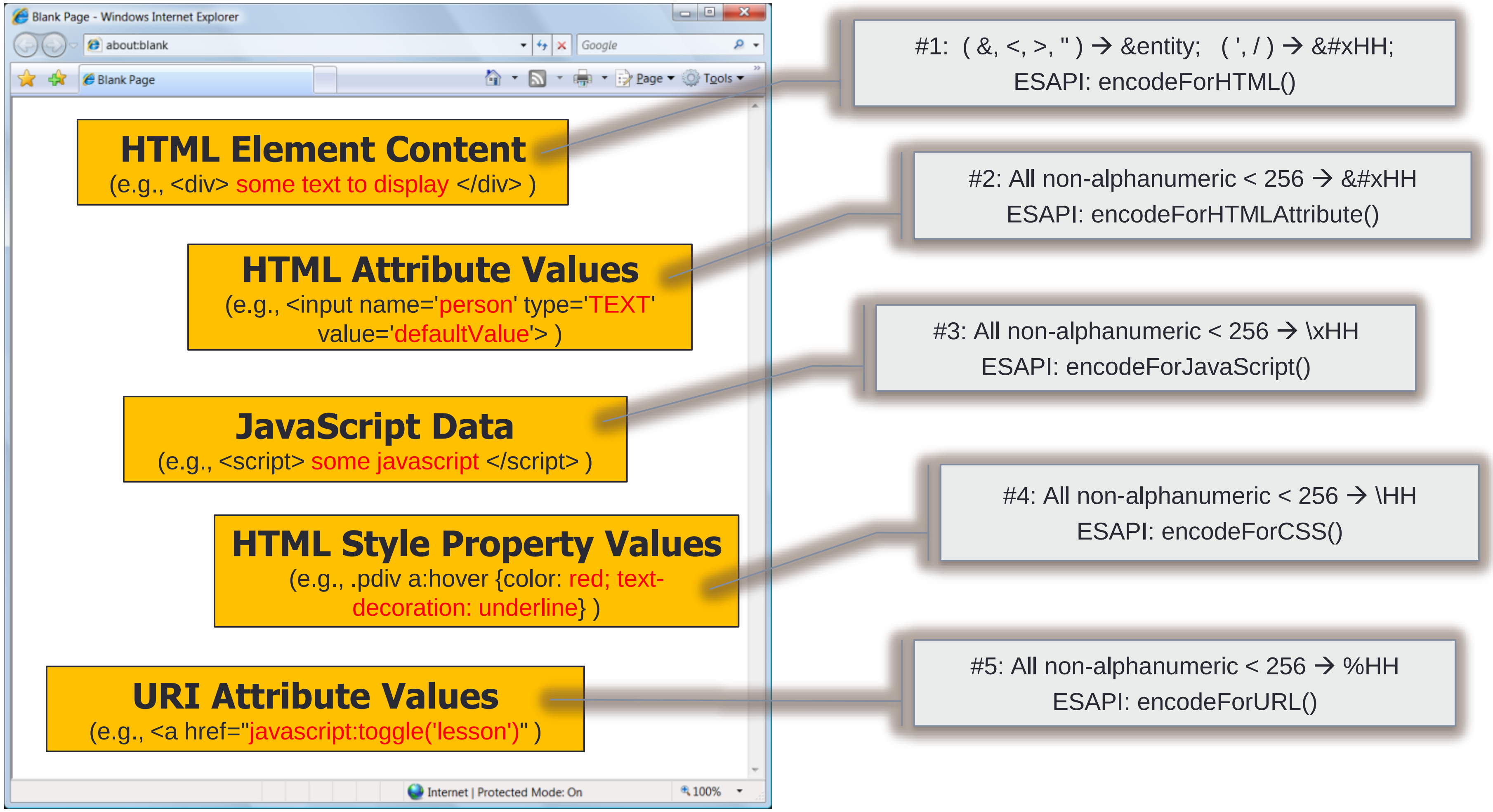
Cross-Site Scripting

- چگونه تشخیص دهیم؟
- ورودی کد جاوااسکریپت
- <script>alert</script>
- آیا کد بر روی مرورگر اجرا می شود؟
- انواع حالت ها می بایست تست شود (در سایت OWASP وجود دارد)

Cross-Site Scripting

- چگونه آسیب پذیری را از بین ببریم؟
- فیلتر داده هایی که توسط کاربر وارد سیستم شده، در هنگام نمایش
- فیلتر داده ورودی
- کتابخانه های در دسترس برای برنامه نویسان
- کتابخانه OWASP ESPI (برای NET.، PHP، جاوا، C/C++)
- OWASP AntiSamy (برای NET. و جاوا)
- Jsoup (برای جاوا یا jsp)
- HTMLPurifier (برای PHP)
- HTTP Only Cookies

نحوه استفاده از ESPI در بخش های مختلف


Blank Page - Windows Internet Explorer
about:blank
Google
Blank Page
Page
Tools
Internet | Protected Mode: On
100%

HTML Element Content
(e.g., <div> some text to display </div>)

HTML Attribute Values
(e.g., <input name='person' type='TEXT' value='defaultValue'>)

JavaScript Data
(e.g., <script> some javascript </script>)

HTML Style Property Values
(e.g., .pdiv a:hover {color: red; text-decoration: underline})

URI Attribute Values
(e.g., <a href="javascript:toggle('lesson')")

#1: (&, <, >, ") → &entity; (', /) → &#xHH;
ESAPI: encodeForHTML()

#2: All non-alphanumeric < 256 → &#xHH
ESAPI: encodeForHTMLAttribute()

#3: All non-alphanumeric < 256 → \xHH
ESAPI: encodeForJavaScript()

#4: All non-alphanumeric < 256 → \HH
ESAPI: encodeForCSS()

#5: All non-alphanumeric < 256 → %HH
ESAPI: encodeForURL()

Cross-Site Scripting

- مطالعه بیشتر
- http://www.owasp.org/index.php/cross-site_scripting
- Owasp cheat sheet
- http://ha.ckers.org/xss.html

A3 – Broken Authentication and Session Management

HTTP is a “stateless” protocol

- Means credentials have to go with every request
- Should use SSL for everything requiring authentication

Session management flaws

- SESSION ID used to track state since HTTP doesn’t
  - and it is just as good as credentials to an attacker
- SESSION ID is typically exposed on the network, in browser, in logs, …

Beware the side-doors

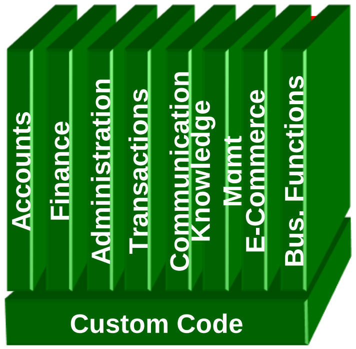
- Change my password, remember my password, forgot my password, secret question, logout, email address, etc…

Typical Impact

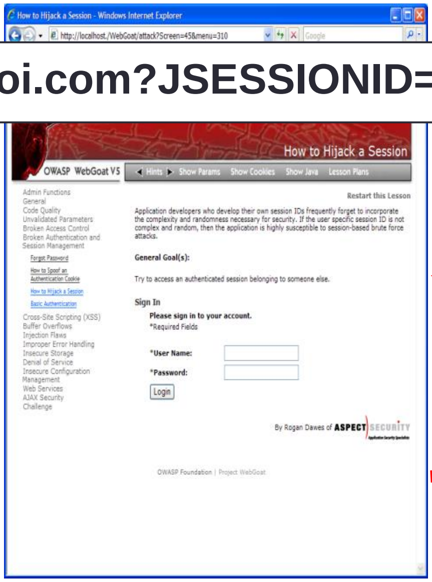
- User accounts compromised or user sessions hijacked

Broken Authentication

1 User sends credentials



www.boi.com?JSESSIONID=9FA1DB9EA...



Application developers who develop their own session IDs frequently forget to incorporate the complexity and randomness necessary for security. If the user specific session ID is not complex and random, then the application is highly susceptible to session-based brute force attacks.

General Goal(s):

Try to access an authenticated session belonging to someone else.

Sign In

Please sign in to your account.
*Required Fields

***User Name:**

***Password:**

Site uses URL rewriting
(i.e., put session in URL) 2

3 User clicks on a link to http://www.hacker.com in a forum

Hacker checks referer logs on
www.hacker.com
and finds user's JSESSIONID 4

5 Hacker uses JSESSIONID and takes over victim's account

A3 – Avoiding Broken Authentication and Session Management

- Verify your architecture
  - Authentication should be simple, centralized, and standardized
  - Use the standard session id provided by your container
  - Be sure SSL protects both credentials and session id at all times

- Verify the implementation
  - Forget automated analysis approaches
  - Check your SSL certificate
  - Examine all the authentication-related functions
  - Verify that logoff actually destroys the session
  - Use OWASP’s WebScarab to test the implementation

A4 – Insecure Direct Object References

How do you protect access to your data?

- This is part of enforcing proper “Authorization”, along with A7 – Failure to Restrict URL Access

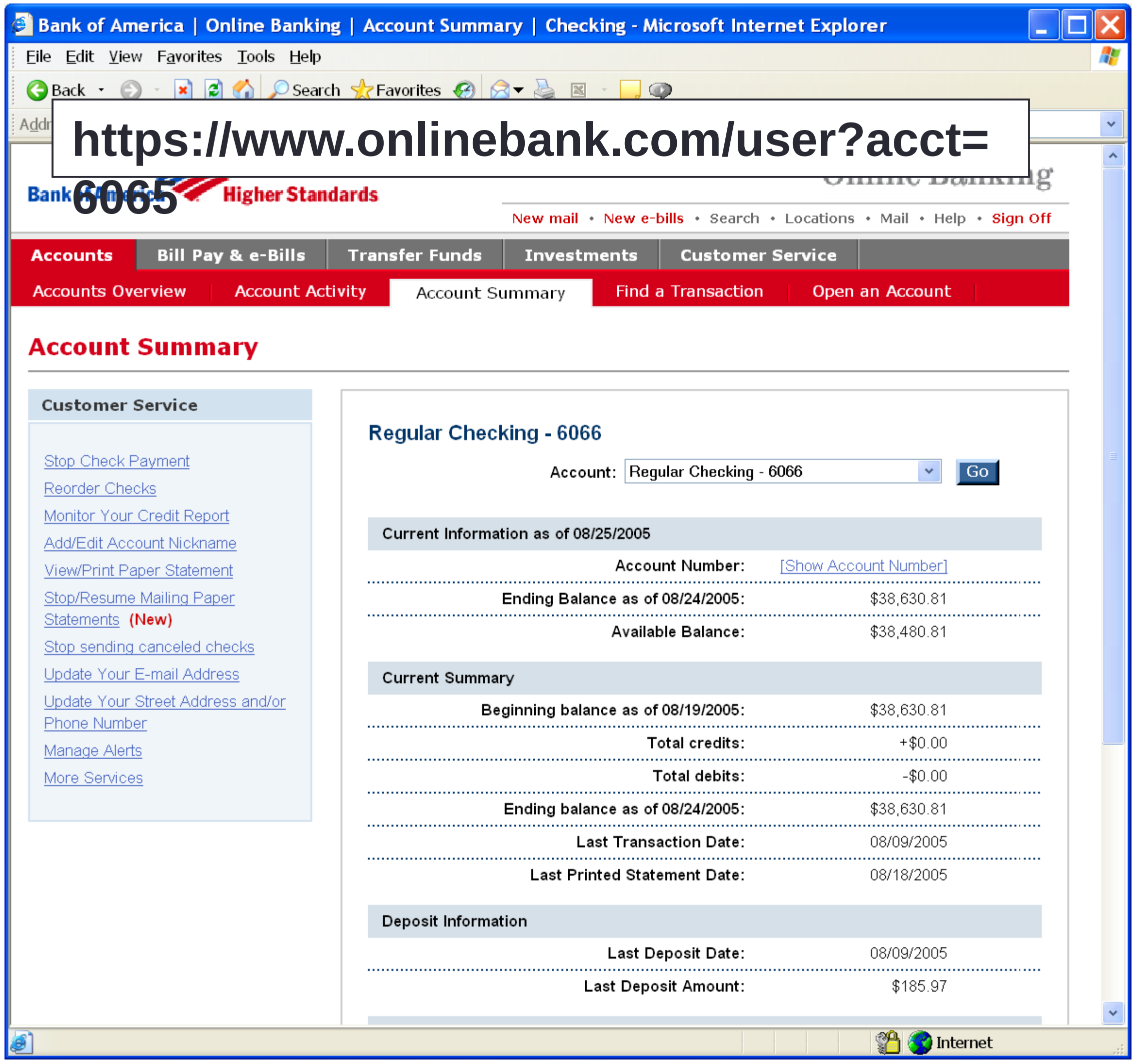
A common mistake …

- Only listing the ‘authorized’ objects for the current user, or
- Hiding the object references in hidden fields
- … and then not enforcing these restrictions on the server side
- This is called presentation layer access control, and doesn’t work
- Attacker simply tampers with parameter value

Typical Impact

- Users are able to access unauthorized files or data

Insecure Direct Object References Illustrated


Bank of America | Online Banking | Account Summary | Checking - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Addr
Bank of America Higher Standards
Online Banking
New mail • New e-bills • Search • Locations • Mail • Help • Sign Off
Accounts Bill Pay & e-Bills Transfer Funds Investments Customer Service
Accounts Overview Account Activity Account Summary Find a Transaction Open an Account
Go
Internet

https://www.onlinebank.com/user?acct=6065

Account Summary

Customer Service

- Stop Check Payment
- Reorder Checks
- Monitor Your Credit Report
- Add/Edit Account Nickname
- View/Print Paper Statement
- Stop/Resume Mailing Paper Statements (New)
- Stop sending canceled checks
- Update Your E-mail Address
- Update Your Street Address and/or Phone Number
- Manage Alerts
- More Services

Regular Checking - 6066

Account: Regular Checking - 6066

Current Information as of 08/25/2005	
Account Number:	[Show Account Number]
Ending Balance as of 08/24/2005:	$38,630.81
Available Balance:	$38,480.81
Current Summary	
Beginning balance as of 08/19/2005:	$38,630.81
Total credits:	+$0.00
Total debits:	-$0.00
Ending balance as of 08/24/2005:	$38,630.81
Last Transaction Date:	08/09/2005
Last Printed Statement Date:	08/18/2005
Deposit Information	
Last Deposit Date:	08/09/2005
Last Deposit Amount:	$185.97

- Attacker notices his acct parameter is 6065

  ?acct=6065

- He modifies it to a nearby number

  ?acct=6066

- Attacker views the victim’s account information

A4 – Avoiding Insecure Direct Object References

- Eliminate the direct object reference
  - Replace them with a temporary mapping value (e.g. 1, 2, 3)
  - ESAPI provides support for numeric & random mappings
    - IntegerAccessReferenceMap & RandomAccessReferenceMap
- Validate the direct object reference
  - Verify the parameter value is properly formatted
  - Verify the user is allowed to access the target object
    - Query constraints work great!
  - Verify the requested mode of access is allowed to the target object (e.g., read, write, delete)

A5 – Cross Site Request Forgery (CSRF)

Cross Site Request Forgery

- An attack where the victim’s browser is tricked into issuing a command to a vulnerable web application
- Vulnerability is caused by browsers automatically including user authentication data (session ID, IP address, Windows domain credentials, …) with each request

Imagine…

- What if a hacker could steer your mouse and get you to click on links in your online banking application?
- What could they make you do?

Typical Impact

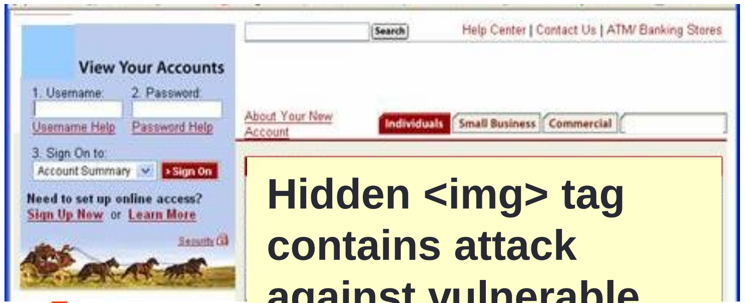
- Initiate transactions (transfer funds, logout user, close account)
- Access sensitive data
- Change account details

CSRF Vulnerability Pattern

- The Problem
  - Web browsers automatically include most credentials with each request
  - Even for requests caused by a form, script, or image on another site
- All sites relying solely on automatic credentials are vulnerable!
  - (almost all sites are this way)
- Automatically Provided Credentials
  - Session cookie
  - Basic authentication header
  - IP address
  - Client side SSL certificates
  - Windows domain authentication

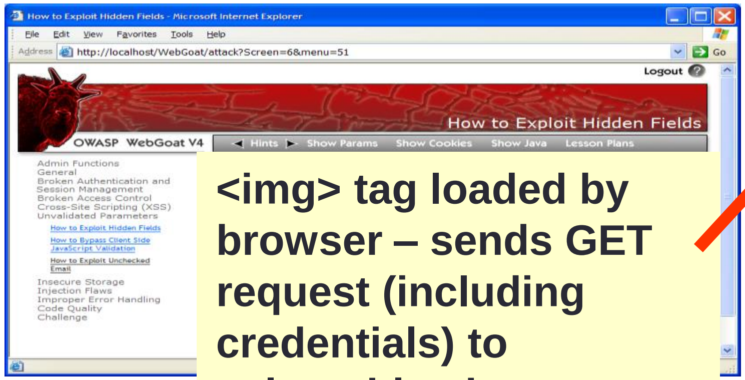
سناریوی نمونه

1 Attacker sets the trap on some website on the internet (or simply via an e-mail)



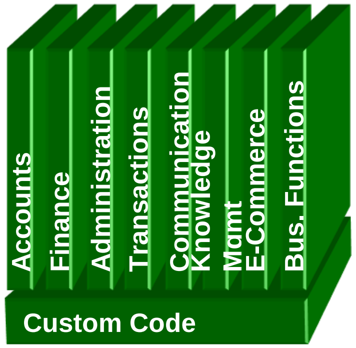
Hidden <img> tag contains attack against vulnerable site

2 While logged into vulnerable site, victim views attacker site


How to Exploit Hidden Fields - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost/WebGoat/attack?Screen=6&menu=51
Go
Logout
How to Exploit Hidden Fields
OWASP WebGoat V4
Hints
Show Params
Show Cookies
Show Java
Lesson Plans
Admin Functions
General
Broken Authentication and Session Management
Broken Access Control
Cross-Site Scripting (XSS)
Unvalidated Parameters
How to Exploit Hidden Fields
How to Bypass Client Side JavaScript Validation
How to Exploit Unchecked Email
Insecure Storage
Injection Flaws
Improper Error Handling
Code Quality
Challenge

<img> tag loaded by browser – sends GET request (including credentials) to vulnerable site

Application with CSRF vulnerability



3 Vulnerable site sees legitimate request from victim and performs the action requested

A5 – Avoiding CSRF Flaws

- Add a secret, not automatically submitted, token to ALL sensitive requests
  - This makes it impossible for the attacker to spoof the request
    - (unless there's an XSS hole in your application)
  - Tokens should be cryptographically strong or random

- Options
  - Store a single token in the session and add it to all forms and links
    - **Hidden Field:** `<input name="token" value="687965fdfaew87agrde" type="hidden"/>`
    - **Single use URL:** `/accounts/687965fdfaew87agrde`
    - **Form Token:** `/accounts?auth=687965fdfaew87agrde ...`
  - Beware exposing the token in a referer header
    - Hidden fields are recommended
  - Can have a unique token for each function
    - Use a hash of function name, session id, and a secret
  - Can require secondary authentication for sensitive functions (e.g., eTrade)

- Don't allow attackers to store attacks on your site
  - Properly encode all input on the way out
  - This renders all links/requests inert in most interpreters

See the new: www.owasp.org/index.php/CSRF_Prevention_Cheat_Sheet for more details

A6 – Security Misconfiguration

Web applications rely on a secure foundation

- All through the network and platform
- Don’t forget the development environment

Is your source code a secret?

- Think of all the places your source code goes
- Security should not require secret source code

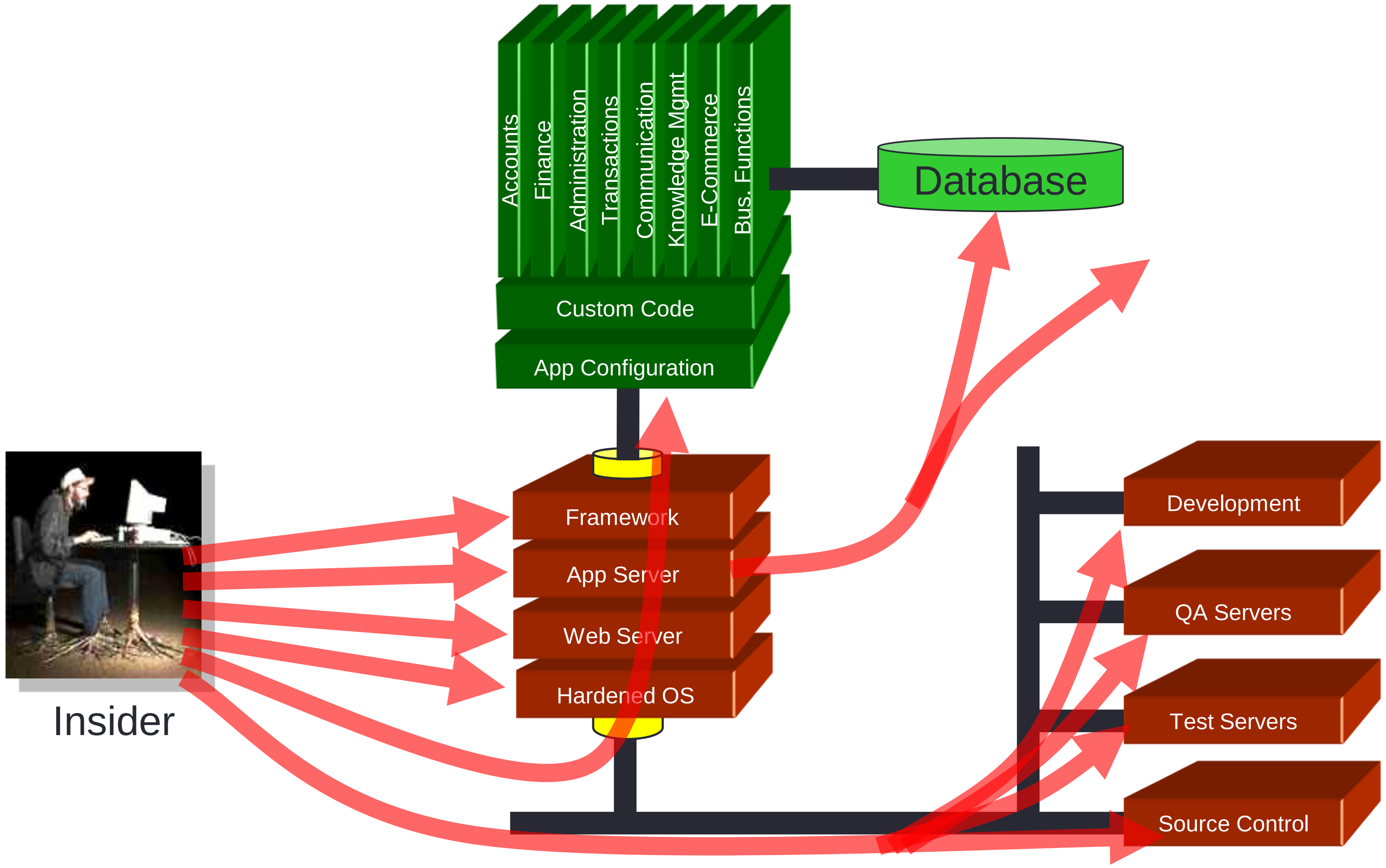
CM must extend to all parts of the application

- All credentials should change in production

Typical Impact

- Install backdoor through missing network or server patch
- XSS flaw exploits due to missing application framework patches
- Unauthorized access to default accounts, application functionality or data, or unused but accessible functionality due to poor server configuration

Security Misconfiguration



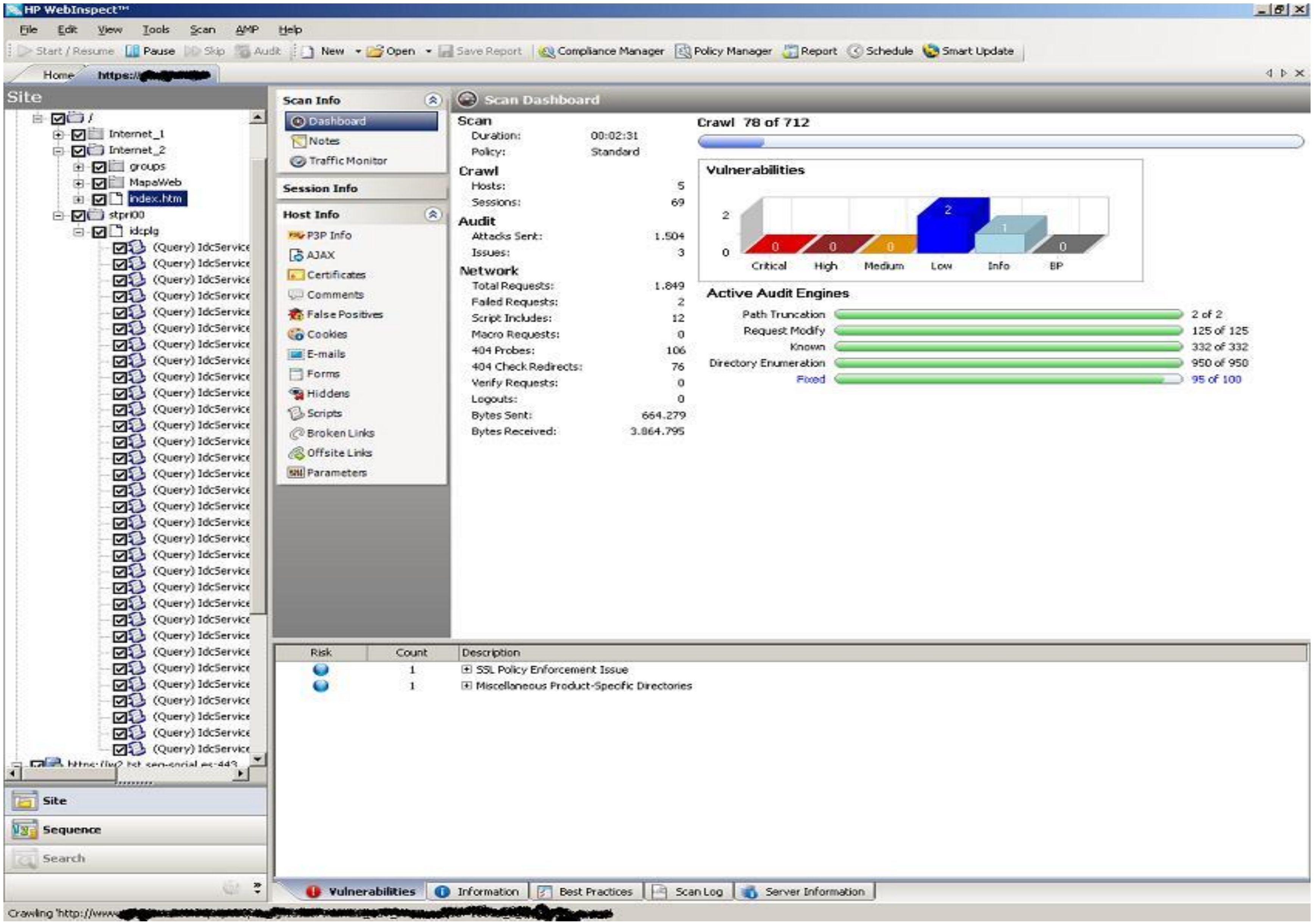
A6 – Avoiding Security Misconfiguration

- Verify your system’s configuration management
  - Secure configuration “hardening” guideline
    - Automation is REALLY USEFUL here
  - Must cover entire platform and application
  - <u>Keep up with patches</u> for ALL components
    - This includes software libraries, not just OS and Server applications
  - Analyze security effects of changes
- Can you “dump” the application configuration
  - Build reporting into your process
  - If you can’t verify it, it isn’t secure
- Verify the implementation
  - Scanning finds generic configuration and missing patch problems

ابزارهای تست

- تجاری
- Accunetix
- Web inspect
- متن باز
- Skipfish
- ZAP Proxy

ابزارها: HP WebInspect



HP WebInspect™

File Edit View Tools Scan AMP Help

Start / Resume | Pause | Skip | Audit | New | Open | Save Report | Compliance Manager | Policy Manager | Report | Schedule | Smart Update

Home | https://[redacted]

Site

- /
  - Internet_1
  - Internet_2
    - groups
    - MapaWeb
    - index.htm
  - stpri00
    - idcplg
      - (Query) IdcService (×32)
- https://lir?.tst.seg-social.es:443

Site | Sequence | Search

Scan Info: Dashboard, Notes, Traffic Monitor

Session Info

Host Info: P3P Info, AJAX, Certificates, Comments, False Positives, Cookies, E-mails, Forms, Hiddens, Scripts, Broken Links, Offsite Links, Parameters

Scan Dashboard

Scan	
Duration:	00:02:31
Policy:	Standard
Crawl	
Hosts:	5
Sessions:	69
Audit	
Attacks Sent:	1.504
Issues:	3
Network	
Total Requests:	1.849
Failed Requests:	2
Script Includes:	12
Macro Requests:	0
404 Probes:	106
404 Check Redirects:	76
Verify Requests:	0
Logouts:	0
Bytes Sent:	664.279
Bytes Received:	3.864.795

Crawl 78 of 712

Vulnerabilities

Critical	High	Medium	Low	Info	BP
0	0	0	2	1	0

Active Audit Engines

Path Truncation	2 of 2
Request Modify	125 of 125
Known	332 of 332
Directory Enumeration	950 of 950
Fixed	95 of 100

Risk	Count	Description
●	1	SSL Policy Enforcement Issue
●	1	Miscellaneous Product-Specific Directories

Vulnerabilities | Information | Best Practices | Scan Log | Server Information

Crawling 'http://www[redacted]