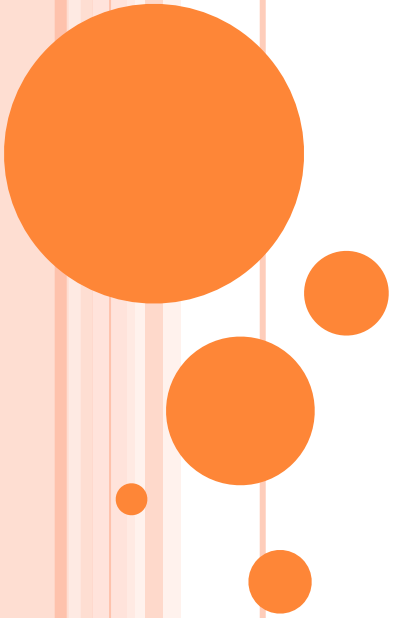


Introduction to Web Security

by Fabrizio d'Amore

Department of Computer, Control, and Management Engineering Antonio Ruberti

Sapienza University of Rome

WHY WEB SECURITY

- recent studies conform a trend that has been observed in last 8 years: strong increase in (quality and number of) threats related to client-server / multi-tier applications, adopted in Web 2.0 applications
- complex scenario, because preventions require study of software components and of their interactions
- critical components are always usually the web/ application server and browser
  - other possible critical components: DBMS server, auth server, etc.

A FEW FUNDAMENTAL PRINCIPLES

source: ***Open Web Application Security Project (OWASP)***

https://www.owasp.org/index.php/Main_Page

Principles (https://www.owasp.org/index.php/Category:Principle)

- Apply defence in depth
  - layered (horizontal) and/or structured (vertical) security mechanisms increase security
  - don't rely on a unique defense (e.g., only border-based defences)
- Use a positive security model
  - based on *white lists*, it prevents by definition new types of attack
- Fail securely (handling errors securely is a key aspect of secure coding)
  - type 1 errors exceptions that occur in the processing of a security control, they **should disallow the operation**
  - type 2 errors exceptions in code that is not part of a security control: they **could affect the way a security control is carried out**

```
isAdmin = true;
try {
  codeWhichMayFail();
  isAdmin = isUserInRole( "Administrator" );
}
catch (Exception ex)
{
  log.write(ex.toString());
}
```

MORE OWASP PRINCIPLES

- Run with least privilege
  - accounts should have the least amount of privilege required to perform their business processes
  - ex.: don't connect to mysql DB as root, you don't need that (normally)
- Avoid security by obscurity
  - it is a weak security control, and nearly always fails when it is the only control
  - instead, use security controls based on open and know principles
  - vulnerabilities come from lack of open principles, not from secrecy
- Keep security simple
  - verifiable, economy of mechanism
  - minimise attack surface (the larger, the more complex)
- Detect intrusions (compromise recording)
  - log security-relevant events
  - ensure logs are monitored regularly
  - properly respond to an intrusion once detected
- Don't trust infrastructure
  - do you know its details?
  - what do you know of its design principles?
- Don't trust services
  - they could refer to external resources
- Establish secure defaults (psychological acceptability)
  - example: password aging enabled by default but users can disable it

THE RISKS (WEB SECURITY)

- server side risks
  - related to system & application configuration
  - related to application code
- client side risks
  - browser related
- risks coming from network
  - at any point of the path client-server

RISKS AT SERVER SIDE

- data stealing
- remote command execution
- collecting data for intrusion setup
- DOS (denial of service) attacks
  - several typologies

RISKS AT CLIENT SIDE

- mainly related to browser
- active contents might cause browser's crash, damage user system, violate privacy…
  - attackers can also execute arbitrary code (botnets)
- use & abuse of private information, obtained from user
  - user can be aware or unaware
  - identity thefts

RISKS COMING FROM NETWORK

- eavesdropping
  - i.e., capture of transmitted data between browser & server by means of electronic interception
- interceptors can operate everywhere in the path between browser & server
  - network at browser's side
  - network at server's side (includes intranet)
- user or server ISP
- ISP's access provider

SECURITY AT SERVER SIDE

- related to applications offering services
- ***Web server*** plays crucial role
- we'll see a few important points
  - correct setting of file access permissions
  - optional services
  - userid of server process
  - shared `ftp/www` areas
  - server in `chroot` environment
  - www server & firewall
  - recognise compromised servers

IMPORTANT DIRECTORIES

- **document root**
  - folder in Web server designated to contain Web pages
  - synonymous: *start directory*, *home directory*, *web publishing directory*, *remote root* etc.
  - typical names of document root: htdocs, httpdocs, html, public_html, web, www etc.
- **server root**
  - contains *logs* & *configurations*
  - a few scripts put here their working directory

FILE PERMISSIONS

- be aware of permissions given to directories
  - document root, containing HTML documents
  - server root, containing log & configuration files; often CGI scripts run here
    - the Common Gateway Interface (CGI) is a standard (RFC3875) that defines how Web server software can delegate the generation of Web pages to a console application. Such applications are known as CGI scripts; they can be written in any programming language, although scripting languages are often used
- good idea: purposely define username and group
  - e.g., www & wwwgroup
  - home directory of www is server root
  - HTML authors should be added to wwwgroup

PERMISSIONS OF SERVER ROOT

- only www should have rwx access everywhere
- wwwgroup could have rx access everywhere and w on contents
- others should have rx access on contents and cgi-bin

```
drwxr-xr-x 5 www wwwgroup 1024 Aug 8 00:01 cgi-bin/
drwxr-x--- 2 www wwwgroup 1024 Jun 11 17:21 conf/
-rwx------ 1 www wwwgroup 109674 May 8 23:58 httpd
drwxrwxr-x 2 www wwwgroup 1024 Aug 8 00:01 htdocs/
drwxrwxr-x 2 www wwwgroup 1024 Jun 3 21:15 icons/
drwxr-x--- 2 www wwwgroup 1024 May 4 22:23 logs/
```

PERMISSIONS OF DOCUMENT ROOT

- since the Web server should run as nobody document root should have permissions o+rx
- in order to allow HTML authors to insert contents document root should have permissions ug+rwx
- same permissions for subdirectories

```
drwxrwxr-x 3 www wwwgroup 1024 Jul 1 03:54 contents
drwxrwxr-x 10 www wwwgroup 1024 Aug 23 19:32 examples
-rw-rw-r-- 1 www wwwgroup 1488 Jun 13 23:30 index.html
-rw-rw-r-- 1 lstein wwwgroup 39294 Jun 11 23:00 tmp.html
```

PERMISSIONS OF DOCUMENT ROOT 2

- many Web servers allow selective access to portions of document root, based on browser's IP or authentication
- however if document root has permissions o+rx local users will be allowed to access all of the contents
- solution: server process runs with id different from nobody, with minimal privileges and belonging to wwwgroup
  - permissions g+rx and o-rwx for "restricted" sections
  - warning: if we use g+rwx also the Web server will be able to overwrite contents

PERMISSIONS OF DOCUMENT ROOT 3

- if there exist specific reasons for making the Web server running with other userid, make sure that log directory is not writeable for such userid
  - otherwise an attacker could exploit a bug in some script and replace a log file with a symbolic link to /etc/passwd: when service is restarted a chown of /etc/passwd occurs and the attacker will be able to add a new line to that file
  - some workarounds exist

OPTIONAL SERVICES

- most Web servers have additional capabilities, that can increase the risk
  - ***automatic directory listing***
  - ***symbolic link following***
  - ***server side include***
  - ***user maintained directory***

OPTIONAL SERVICES / DIRECTORY LISTING

- knowledge is power
  - backup files left by text editors, maybe with scripts source code
  - temporary directories and files
  - symbolic links, that have been created for convenience, but that have never been removed
  - source code control logs
- disabling the directory listing does not prevent from guessing file names
  - all unnecessary files should be carefully removed

OPTIONAL SERVICES / SYMBOLIC LINK FOLLOWING

- extending the document tree with symbolic links could be useful, but awkward actions could make reachable delicate directories, such as /etc
  - apache can be configured to allow following of symbolic links, but only when link owner and destination owner do coincide, so that possible damages are limited to such owner
- another approach allows to extend the document tree directly acting on the configuration (e.g., alias of apache)

OPTIONAL SERVICES / SERVER SIDE INCLUDE

- Server Side Includes (SSI) is a simple interpreted server-side scripting language used almost exclusively for the Web
- it is mainly used for dynamically including the content of a file inside another file, while the latter is being served by a Web server
- in addition to inclusion, there is the **exec** directive that allows the execution of programs, scripts, command shells etc. at server side
  - potential vulnerability

SERVER SIDE INCLUDE

- SSI-enabled files have usually the extension .shtml, although this is not really needed
- syntax

  <!--#directive parameter=value parameter=value -->

- notice the use of HTML comment, useful when SSI is disabled
- the output of the command is inserted into the file being served from the Web server, in place of the command itself
- main directives: include, exec, echo, config, flastmod, printenv

SERVER SIDE INCLUDE 2

- often used for the automation of simple tasks
  - more complex tasks use Perl, PHP, ASP, Python, Ruby, JSP etc.
- also, client side includes exist, obtained at client side through frames, IFrames, JavaScript etc.
- see for instance http://en.wikipedia.org/wiki/Server_Side_Includes

OPTIONAL SERVICES / USER-MAINTAINED DIRS

- many communities welcome this optional service, that allows users to automatically add to document root personal portions of file system
  - the well-known path ~userid of apache
- it is a potential breach in security, due to clumsy actions that users could do
- if such a service is activated it is even more urgent to disable SSI and symbolic link following

USERID OF SERVER PROCESS

typical scenario

- the Web server is run with userid root so that it can open port 80 and write log files
- when a connection on port 80 comes, the process makes a fork for serving the request
  - the child has userid nobody (or similar)
- this is a robust scenario

USERID OF SERVER PROCESS 2

vulnerable scenario

- child receives userid root
  - it is sufficient to specify "User root" in the configuration file
- every script will run with root privileges

USERID OF SERVER PROCESS 3

alternative scenario

- not even parent is root
- consequences
  - it won't be able to open port 80 (Unix), neither other well-known port; it will have to use a high port number (e.g., 8000, or 8080)
  - configuration files must be made readable by parent
  - logs must be readable and writeable by parent
  - scripts containing errors, or altered, will be able to access read/write such files

SHARED AREAS WWW/FTP

- in some cases, it is appreciated the (partial) overlapping of www & ftp areas
- this is fine as long as remote users cannot upload a file (via ftp) that can be read or executed by the Web daemon
  - for instance, it suffices to make incoming not readable to nobody
  - otherwise, the attacker will upload a .cgi file

CHROOT ENVIRONMENTS

linux system command chroot

chroot /path/to/new/root /server_root/httpd

CHROOT(1) User Commands CHROOT(1)

NAME

chroot - run command or interactive shell with special root directory

SYNOPSIS

chroot NEWROOT [COMMAND...]

chroot OPTION

DESCRIPTION

Run COMMAND with root directory set to NEWROOT.

CHROOT ENVIROMENTS 2

- a tiny file system has to be created, where the Web server can find all resources it could need:
  - special files (devices), shared libraries
  - configuration files adapted to the new path
- a bare environment will make the approach effective
  - no interpreters, shells, passwords and other configuration files in the tiny file system (configuring interpreters will re-introduce vulnerabilities, albeit to a smaller extant)

WWW SERVER & FIREWALL

- make the server a “sacrificial lamb”, and put it outside the area protected by the firewall
  - the server can be captured, but the intranet (or LAN) will remain intact
- *do not* install www server and firewall in the same host
  - compromised www ⇒ compromised firewall ⇒ compromised intranet
- more complex architectures exist, that require both internal and external servers

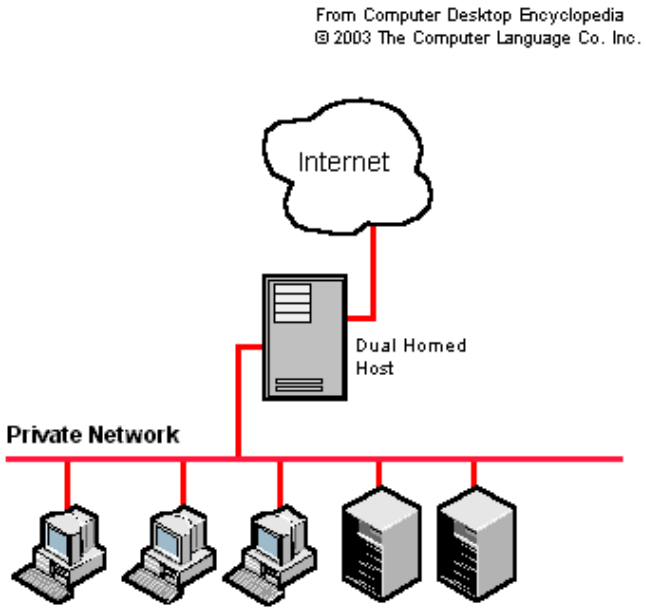
WWW SERVER & FIREWALL 2

problem

- some firewalls does not allow positioning a host outside the perimeter
- a passage is to be opened in the firewall, in a way depending upon firewall's technology
  - **dual homed host**
  - **screened host**
  - **screened subnet**
  - **application-level gateway**
  - **bastion host**

DUAL HOMED HOST

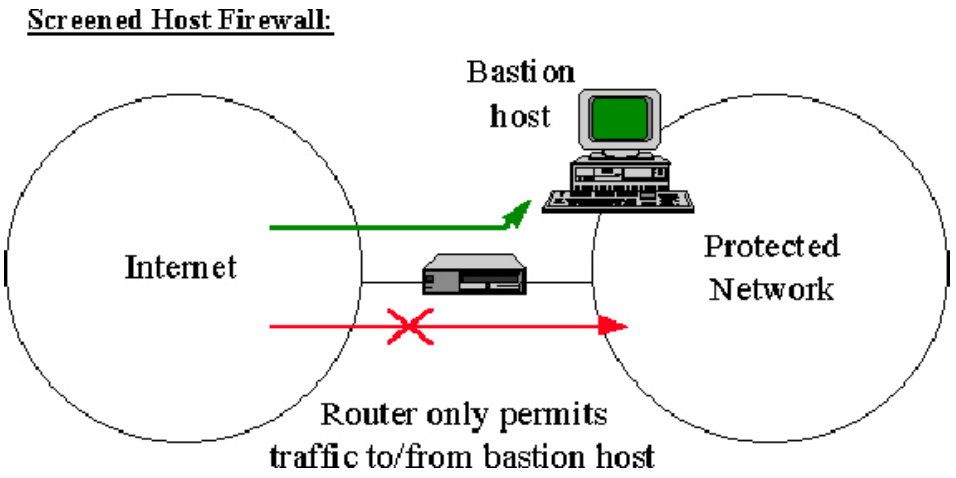
- host having two network adapters, not routing packets
- implements the insulation of the two network segments
- users of both networks can use applications installed on host



- routing is done at applicative level: application can receive packets from a network adapter and decide whether routing them towards the other network adapter

SCREENED HOST

- a *screening router* forwards all inbound traffic to a *bastion host* and only allows the bastion host to send packets outside
  - it is possible to select a few hosts of the protected network and allow them direct external access
- in practice, a proxy server on bastion host is enforced



BASTION HOST

- unique host that is reachable from the Internet
- massively protected host
- secure operating system (hardened or trusted)
- no unneeded software, no compilers & interpreters
- proxy server in a insulated environment (chrooting)
- read-only file system
- process checker
- integrity file system checker
- small number of services and no user accounts
- untrusted services have been removed
- saving & control of logs
- source-routing disabled

WWW SERVER & FIREWALL 3

passage in

- dual homed host
  - in practice, a proxy has to be installed on the firewall
- screened host
  - allow (bastion host or screening router) direct forwarding of packets going to / coming from port 80

RECOGNIZING A COMPROMISED SERVER

- use a *host-based intrusion detection system*
  - software system controlling and analyzing the state of a computing system
  - e.g., Tripwire (commercial), OSSEC (multi-platform, open source, but purchased at May 2009 by Trend Micro, with promises to keep the project open source and free); see also http://en.wikipedia.org/wiki/Host_based_intrusion_detection_system
- periodically check log files (both accesses & errors) and look for
  - system commands such as rm, login, /bin/sh, perl
  - very long URLs (*buffer overrun*)
  - repeated and failed attempts to guess passwords

COMPROMISING CLIENT SIDE

- attackers try to exploit vulnerabilities on clients for controlling the host, capturing data, deleting files and using host for attacking other hosts
- this can be easily done by exploiting browsers' vulnerabilities
  - attackers can create a Web page that will install Trojan or spyware
  - instead of actively attacking vulnerable systems, a malicious Web site can passively compromise a system when receiving a visit
  - malicious HTML documents can also be sent via e-mail to victims

Trojan horse

- apparently useful application or tool, containing hidden features that can exploit user privileges, thus threatening security
- often installed by (unaware) users or intruders who got access thru other means
- intruders trying to compromise a system thru Trojan horses base their success upon users running the Trojan
- fundamental concepts and famous incidents: http://www.cert.org/advisories/CA-1999-02.html

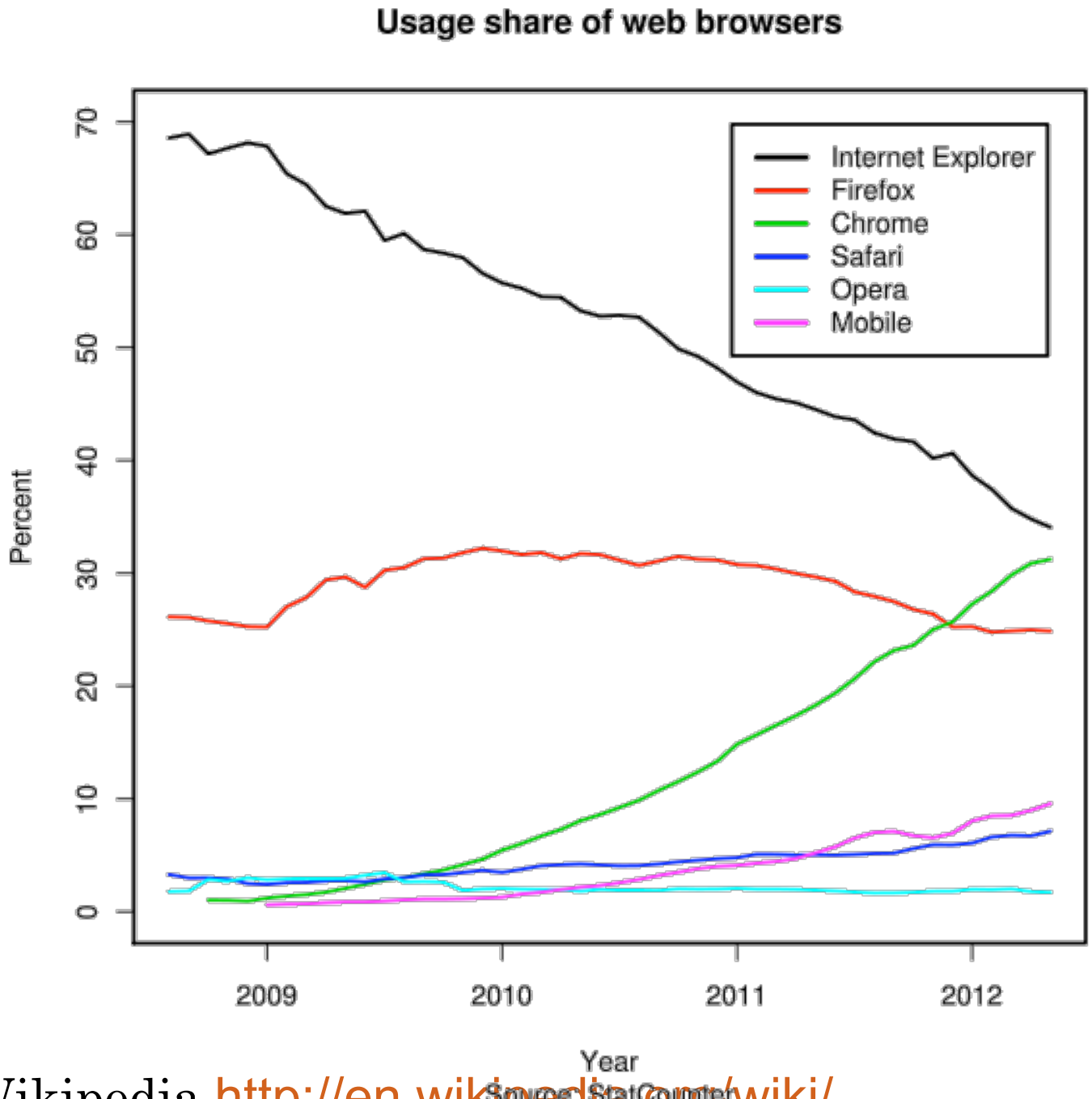
SPYWARE

- software collecting data of a computing system without owner's consent
- spyware exists since '80 (first keyloggers)
- collected data are often sent to subjects interested in selling data or in direct usage, for purposes like financial crimes, identity theft, marketing and spam
- technically speaking the term spyware holds when the explicit consent to data collection and/or communication is missing
- in some cases, software licenses are written in a tricky and unclear way, for inducing the user to accept the proposed conditions, making him subject to some type of legal spyware

SPYWARE 2

- who is spying us?
  - online attackers
  - organized crime
  - marketing organizations
  - trusted insider
- how?
  - hijacking of browsers' sessions
  - browser helper objects (BHO)
  - cookies and web bugs
  - false anti-spyware tools
  - autonomous spyware
  - bots
- what do they gather?
  - Internet activity
  - e-mail and contact information
  - Windows Protected Storage data (until Windows Server 2003)
  - clipboard contents
  - keystrokes
  - screenshots
  - network traffic

http://www.cert.org/archive/pdf/spyware2005.pdf



from Wikipedia http://en.wikipedia.org/wiki/Usage_share_of_web_browsers

VULNERABILITIES IN BROWSERS

- strong recent increase of threats related to vulnerabilities in browsers, thru compromised or malicious Web sites
- in principle, users should be able to evaluate risks related to software usage

problem

- many computers are sold with pre-installed and pre-configured software
- average user doesn't want or can't understand how installed software can interact

AGGRAVATING CIRCUMSTANCES

- users don't really check what links they are going to click
- Web pages can link disguised URLs
- unexpected redirections
- new functionalities in browsers (there is a browsers' war!) at the cost of lower security
- new vulnerabilities discovered after software has been configured and packed by the producer
- additional software too easily installed
- third party software could not include a software update subsystem
- Web sites may require enabling specific functionalities and/or installing additional software
- many users are not capable of setting up their browser in a secure way
- many users don't want to enable/disable browser features for making it more secure

BROWSERS: ADDITIONAL FUNCTIONALITIES AND RISKS

- enabling additional features increases the risk
  - the "exposed surface" is wider
  - a few functionalities are enabled by default
- browsers offer similar functionalities but can be based on deeply different technologies
  - many effective technologies have been exported/ imported between different browsers

FREQUENTLY USED TECHNOLOGIES (FUTS)

- *ActiveX*
- *Java*
- *plug-ins*
- *cookies*
- JavaScript
- VBScript

ACTIVEX

- used by Microsoft Internet Explorer on Windows hosts
- the ActiveX technology allows the browser to use applications or parts of them
- ActiveX components can be downloaded from the Web site or can be already installed on the host
- not a programming language, but extension of a (predisposed) application
- evolution of former technologies
  - OLE ⇒ COM ⇒ ActiveX

ACTIVEX 2

- ActiveX controls are files containing commands & functions
- Internet Explorer can import their controls, exploiting their power
- recent further evolution in .NET
- can access the whole Windows system
- integrates advanced functions, but may expose users to attacks and unwanted software installations
- digitally signed but no way to know in advance what they will install

ACTIVEX 3

- when installing application, ActiveX controls might be installed and they can be used by Internet Explorer even if they were not designed for that purpose
- last versions of IE decrease the risks related to ActiveX
- in 2000 CERT/CC organized a workshop on the security of ActiveX
  - http://www.cert.org/reports/activeX_report.pdf
- There exists a US-CERT *Vulnerability Notes Database for ActiveX*: http://www.kb.cert.org/vuls/byid?searchview&query=activex

JAVA APPLETS

- applets are provided by Web sites and executed by a JVM instanced by the browser
- in principle, less dangerous than ActiveX because they are run inside a sandbox
  - but there exist buggy and vulnerable implementations of sandboxes
- There exists a US-CERT *Vulnerability Notes Database for Java*: http://www.kb.cert.org/vuls/byid?searchview&query=java

PLUG-INS

- applications that have been thought for use in browsers
- NPAPI is a standard developed by Netscape and adopted by Firefox, Safari, Opera, Konqueror, Chrome etc. for plug-ins development
- similar to ActiveX controls but cannot be executed outside the browser
  - e.g., Adobe Flash
- can be subject to techniques like buffer overflow, or can exhibit cross-domain violations when the "same-origin" policy is not complied
  - documents/scripts obtained from a certain origin cannot receive/give properties of documents of other origin
  - according Mozilla, two pages have same origin if they have same protocol, port and host

"SAME ORIGIN" (WIKIPEDIA)

Origin determination rules [edit]

The term "origin" is defined using the domain name, application layer protocol, and (in most browsers) port number of the HTML document running the script. Two resources are considered to be of the same origin if and only if all these values are exactly the same. To illustrate, the following table gives an overview of typical outcomes for checks against the URL "http://www.example.com/dir/page.html".

Compared URL	Outcome	Reason
http://www.example.com/dir/page.html	Success	Same protocol and host
http://www.example.com/dir2/other.html	Success	Same protocol and host
http://www.example.com:**81**/dir/other.html	Failure	Same protocol and host but different port
https://www.example.com/dir/other.html	Failure	Different protocol
http://**en.example.com**/dir/other.html	Failure	Different host
http://**example.com**/dir/other.html	Failure	Different host (exact match required)
http://**v2.www.example.com**/dir/other.html	Failure	Different host (exact match required)

COOKIES

- files that are put on client for storing data useful to specific Web sites
  - saved data decided by the Web site and may include information on visited sites or even credentials for accessing the site
- a cookie is only readable to the Web site that has created it
  - when visiting again a Web site browser sends back unchanged cookies
- two kinds of cookies
  - ***session cookies***, removed when browser quits
  - ***persistent cookies***, staying on client until an expiration date (defined by Web server)

COOKIES 2

- they were originally introduced for implementing shopping carts
- used by Web servers for differentiating users and maintaining user data during their navigation, possibly covering many visits
  - allow to recognize users that have been already authenticated
  - permit to customize pages, e.g., the number of results per page returned by Google, or a preferred skin
  - allow to track visitors for a given site, to the purpose of building visiting statistics
  - also implement multi-domain tracking, for modeling the behaviors of users (that remain anonymous) to the purpose of showing personalized advertising, adherent to their profiles

COOKIES 3

- supported by all browsers, that provide a cookie manager
- can be enabled/disabled
  - can also set an inspection mode, for deciding about each cookie
- JavaScript command

```
javascript:alert("Cookies: " + document.cookie)
```

for visualizing active cookies w.r.t. a given page

PRIVACY AND THIRD-PARTY COOKIES

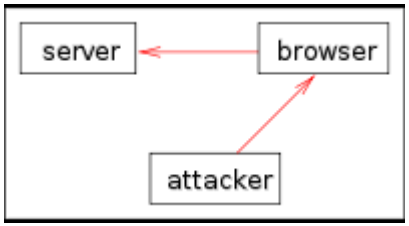
- a Web page can host contents coming from other Web servers
- cookies that are sent by these servers are named ***third-party cookies***
- there are organizations operating in the advertisement that use third-party cookies for tracking users across different sites
  - tracking obtained by pages with ads or with ***web bugs*** (small invisible images sent by third-parties)
  - this allows ads consistent to user profile
- doubts about privacy preserving!

PRIVACY AND THIRD-PARTY COOKIES 2

- user profiling is considered as a potential threat to user privacy, both in one and (specially) in many domains
- several countries have issued laws on the topic
- Directive 2002/58/EC of the European Parliament and of the Council of 12 July 2002 defines rules on cookies: data can be stored on user's computer only if
  1. user has been informed about the use of such data
  2. user can block the operation
- saving data for technical reason is however allowed

PROBLEMS WITH COOKIES

- inaccurate identification
  - many browsers, many storage areas
  - many users using same account, same storage area
- cookies hijacking, for capturing private data
  - based on packet sniffing
  - SSL/TLS can prevent hijacking, indeed https is often used for critical steps such as authentication
- cookies poisoning
  - attacker alters cookie contents (e.g., total paid)
- cross-site cooking
  - attacker exploits user's browser for sending invalid cookie to web server



- client/server misalignment
  - inconsistency due to (possibly repeated) use of "back button" of browser
- cookies expiration
  - expiration date can be in the remote future (privacy violation!)
  - effects of cookies used for session hijacking are persistent

SUPER COOKIES

- cookies that have been stored in computer by means of non-traditional methods
- they are persistent
- browsers don't yet support super-cookies management
  - not even Firefox and IE
  - add-ons start to be available (e.g., BetterPrivacy for Firefox)

FLASH LOCAL SHARED OBJECTS

- from Wikipedia: *"A Local Shared Object (LSO) is a collection of cookie-like data stored as a file on a user's computer. LSOs are used by all versions of Adobe Flash Player and Version 6 and above of Macromedia's now-obsolete Flash MX Player"*
- LSOs can be blocked by defining, site by site, the behavior of Flash
- there is a *Global Storage Setting Panel* available at http://www.macromedia.com/support/documentation/en/flashplayer/help/settings_manager03.html
  - unsatisfactory solution that might make it failing legitimate sites

JAVASCRIPT

- super cookies can be implemented as Javascript programs stored in browser cache
  - browsers have different behaviors
  - IE: chronology, favorites, XML files and saved HTML files
  - other browsers: stored into memory, data persist during navigation
- not easy finding ad hoc solutions
  - disabling Javascript is unsatisfactory
  - can use add-ons (NoScript), that allow limited control
  - can configure browser to empty cache on quit

JAVA

- similar to Javascript
- in Windows it is possible to configure the behavior of Java:
  - "Java" Control Panel, disable "store temporary files on computer" (active by default)
- in Linux something similar is also possible, depending upon what distribution and what Java support has been chosen
  - e.g., Ubuntu (Sun Java): “System” – “Preferences” – “Sun Java 6 Plugin Control Panel”

USERDATA

- only IE
  - see http://msdn.microsoft.com/en-us/library/ms531424(VS.85).aspx
- used also by WindowsUpdate
- can be disabled by customizing the Internet Protection in the Control Panel "Internet Options"

DOM STORAGE

- alternative to cookies, considered more secure
  - supported by Firefox, IE, Chrome and Safari
- it stores pairs (key, value) in a secure setting for future use
- modern interactive Web applications use DOM Storage
- can be disabled but legitimate applications can misbehave
  - e.g., thru about:config in Firefox
- possible approach: destroy data at browser start/ end (e.g., BetterPrivacy add-on)
- see: https://developer.mozilla.org/En/DOM:Storage