Assignment - PSO 6

Purdue CS426 - Computer Security - Prof. Spafford

Harris Christiansen - April 22, 2018
christih@purdue.edu

Read the following:

- Get vs. Post - https://www.youtube.com/watch?v=UObINRj2EGY
- HTTPWatch - http://blog.httpwatch.com/2009/02/20/how-secure-are-query-strings-over-https/
- Session Cookie (note random number is Session ID) - https://www.youtube.com/watch?v=90PsaIatDYo
- Stored XSS - https://www.youtube.com/watch?v=7M-R6U2i5iI
- Reflected XSS - https://www.youtube.com/watch?v=V79Dp7i4LRM
- HttpOnly Cookie - https://stackoverflow.com/questions/14691654/set-a-cookie-to-httponly-via-javascript
- Cross Site Request Forgery - https://www.youtube.com/watch?v=vRBihr41JTo
- SQL Injection - https://www.youtube.com/watch?v=_jKylhJtPmI
- https://www.youtube.com/watch?v=ciNHn38EyRc
- OWASP Top 10 Vulnerabilities - https://www.owasp.org/index.php/Top_10-2017_Top_10
- NMAP - https://www.networkcomputing.com/networking/nmap-tutorial-common-commands/520799832
  https://hackertarget.com/nmap-cheatsheet-a-quick-reference-guide/
- TCP Dump - https://www.tcpdump.org/tcpdump_man.html
- RAW Sockets - http://opensourceforu.com/2015/03/a-guide-to-using-raw-sockets/

Problem 1

(60 pts.)

What are HTTP Cookies in a few sentences?

- HTTP Cookies are pieces of data (key:value) which are stored on the client device by the server. Key:value data (cookies) can be set by the server in the HTTP header of any response to the client. Cookies are sent by the client with requests to the server. Cookies can optionally have additional parameters, including expiration time (after the expiration time the cookie is invalid and the data is discarded).
- Cookies are useful for the server to store user related or identifying data on a client, which can be used to identify, authenticate, validate, or update user data upon requests.

They are commonly used for remembering client side stateful information, and recording users activity.

How can HTTP Cookies be used to create a stateful connection with a webserver?

- Cookies are not encrypted, and can read/modified by the client. Thus, they are not secure. To utilize cookies for purposes such as authentication, a secure auth token should be generated by the server and stored in the cookie (opposed to storing username and password in the cookie), and the auth token should be verified by the server as being valid and likely to belong to the host that made the request.
- Cookies can also be used to support session data, where client data is stored on the server, and a session ID is sent back to the client and stored in a cookie. This session ID is then included with future requests, and is used to lookup the session data that corresponds to the client. Sessions are also not entirely secure, as it is possible to obtain another clients session ID and thus gain access to their session.

When a server side app responds with a session cookie what is the minimum information that is usually stored in the cookie sent to the client machine?

- Continuing from above, when the server sets a session cookie, all that is stored is a unique ID which can be used by the server to lookup the session data (which is stored on the server).
- Session cookies typically include an expiration date, after which the session is no longer valid and data is deleted. This expiration is also enforced server side, and the session ID will no longer be valid after that time. Session cookies are usually updated (renewed) upon every request, such that they only expire after the user becomes inactive.
- An example of a session cookie is the following:
  - Key: site_session
  - Value:
    eyJpdiI6IkVDRXVDakF6MGp3XC82VXp0VitEV0RBPT0iLCJ2YWx1ZSI6Ik1DMGpL
    QjZMeGZBakhlWkJsZHJ2dmN0UkgzWmxhY0FhYjZ2U3h3WmJ1Rmw2ckgrREZM
    WHUrYStHM2xmVTZqT2dYSVwvNjI5YjdOejNhS2F5bVZCZ3pWQT09IiwibWFjIjoi
    NTU0ZjEyYTYzMmI5ZmYzMDZiYTVjYWIyZmRkOTM2OGZjMGVmMGVlNDZjM
    2YxZDVmZWI2OWEyZDhiNDkwM2ZmYyJ9
  - Expires: 4/24/2018

What could potentially happen if an attacker gets access of your cookie that contains session information (i.e. session id)? Explain two ways this risk is mitigated.

- If an attacker obtains another users session id, that attacker can impersonate that user via requests to the server with the users session cookie. This would then give the attacker that users session, often including authenticating them and revealing private information.
- This risk can be mitigated by:
  - The client and server using https, so the cookie cannot be sniffed.
  - The server performing additional validation on the request containing the cookie, including verifying the origin IP/address and other headers (such as plugins installed, browser, and OS - data which can often be used to identify a client) match those originally sent by the client that authenticated and created the session.

Problem 2

Is it a good idea to store sensitive information in a cookie sent back to a client device? Explain. (20 pts.)

- No. Because cookies are stored unencrypted and can be read/modified by the client, information stored in a cookie should never be treated as secure.
- Because cookies cannot store data securely, they should not be used to store sensitive information.
- Instead, techniques such as sessions should be used to keep sensitive information on the server, while still associating it with the client.

Problem 3

What is Stored Cross Site Scripting (XSS) and where is the malicious code run (i.e. web server or client machine)? Explain how the exploit works and how you can mitigate the risk as a software developer and user. (40 pts.)

- Cross site scripting is a method of attack where a malicious client can make a request to a server with malicious code, and that malicious code is later returned to other clients of the server. The malicious code is then executed by those other clients of the server.

- In Stored XSS, the malicious code is sent to the server by a client, and is then stored on that server, and later returned to other clients. This method of attack only works on sites that show users posts to other users, and do not validate the message to ensure it is not XSS.
- For example: On a blog, forum, or other user content driven website, a malicious user may make a post (thus storing their message on the server) with a message containing malicious code, which will be returned to other users who view the post. This malicious message could be a html element, such as an alert which pops up in front of the user, or an <img> which gets rendered on the client (and also makes a request to the server of the image src url).
- To mitigate the risk as a developer:
  - Validate requests coming into the server, including all get and post parameters. Write validators, and reject any requests containing invalid parameters.
  - Encode html entities and escape sql queries. This would result in malicious code no longer being executed. For example: <img src="http://maliciousssite.com"> -> <img src="http://maliciousssite.com">
- To mitigate the risk as a user:
  - Use up to date browsers which have built in protection for XSS attacks.
  - Disable javascript.
  - Manually verify html responses for potential XSS injections into messages, or verify every outgoing html request for unexpected requests.
  - Use secure websites that do not have XSS vulnerabilities.

Problem 4

What is Reflected Cross Site Scripting (XSS) and where is the malicious code run (i.e. web server or client machine)? Explain how the exploit works and how you can mitigate the risk as a software developer and user. (40 pts.)

- Reflected XSS is another XSS attack where an attacker can cause malicious code to be executed on other users devices.
- Instead of malicious code stored on the server being returned to other clients, the reflected XSS vulnerability exists in servers that return data from a clients request in their response to that same client.

- This vulnerability makes it possible for a client to make a request with data such as an html element or javascript, and then have that code be returned in the response and executed by the browser.
- Attackers can use this vulnerability to send out malicious links in the hopes of users clicking those links and being affected by the malicious code.
- To mitigate this risk as a developer: Do not reflect elements from a clients request in any response. If you do, properly validate, escape, and encode the data so it cannot be exploited.
- To mitigate this risk as a user:
  - Do not click on any potentially malicious links.
  - Monitor your outgoing requests for suspicious activity, as reflected XSS attacks could be triggered without you knowing (for example an img url that was posted on another site)
  - Use secure websites that do not have XSS vulnerabilities.

Problem 5

What is Cross Site Request Forgery and where is the malicious code run? Are web pages that use GET or POST request for forms more or less susceptible to the attack? Explain how the exploit works and how you can mitigate the risk as a software developer and user. (60 pts.)

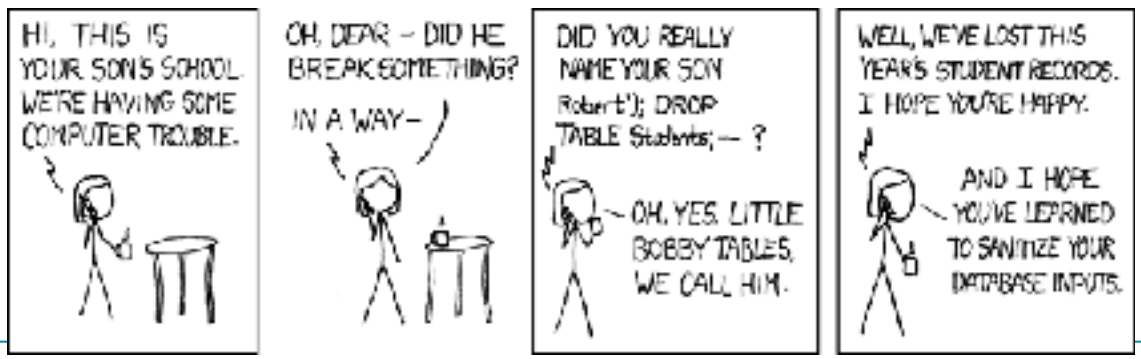
- A Cross Site Request Forgery (CSRF) is a request made from a clients device from another website or service on the clients device.
- Both get and post requests can be made for any website by any other website, service, or program on the clients device.
- CSRF requests are commonly used for various different purposes, such as submitting a form or scraping data from other services. However, they can also be used maliciously.
- A malicious CSRF attack involves a malicious website or service automatically performing a get or post request to a target website (such as your bank), which will be sent using your cookies and authentication. Thus, this request will be authenticated, and can perform any action you are authorized to, including transferring money, purchasing a product, etc.
- To mitigate this risk as a developer:
  - Use CSRF tokens: A CSRF token is a one-time token that is included in the original form/request, and used to validate the request. If the CSRF request does not contain

the CSRF token, or the CSRF token is incorrect, the request will fail. Modern security practice is to include CSRF tokens for all post, put, delete, etc requests.
    - Use user tracking / analytics to verify if the user was on a path where they are likely to make that request. For example, if the user was last on the home page 1 hour ago and never went to the send payment page, it is unlikely they are attempting to send a payment.
- To migrate this risk as a user:
    - Only use trusted websites/programs.
    - Use secure services that protect against CSRF attacks.
    - Log out of important services so your device is no longer authorized.
    - Monitor your outgoing requests for suspicious activity / unexpected requests.

Problem 6

What is a SQL Injection Attack and where is the malicious code run? Explain how the exploit works and how you can mitigate the risk as a software developer and user. (40 pts.)

- An SQL injection attack is an attack aimed at compromising the application database on the server. The malicious code is sent by an attacker in a request, and executed by the server on a database.
- Web applications commonly use databases to store their application and user data. These databases are interacted with using SQL commands. Sometimes these commands include data sent from the user by a get or post request. For example, a user may request to load the profile for user `7`, or they may submit a new post with a large message contents.
- An SQL injection attack attempts to modify the sql query via these user provided fields. For example, an attacker could send a request with the data `Robert'); DROP TABLE Students; —`, which would still result in a valid SQL query, but would perform additional commands unintended by the server.



- Any single SQL injection vulnerability constitutes a complete security breach, and all data in the database should be considered leaked.
- To mitigate this risk as a developer:
  - ALL queries should be properly built, sanitized, and escaped, so that user-provided fields cannot have unintended consequences on the command. Many mysql libraries provides some basic builtin protection, such as mysql_real_escape_string, which should be used around user-provided data.
  - The server should also perform SQL queries using the lowest applicable permissions. For example, queries which only lookup data (and don't update) should be executed with a mysql user that only has permission to read data. Likewise, sql queries that update data should only have permission to read/update, not delete data.
- To mitigate this risk as a user:
  - Only use secure websites and services, which are known not to have SQL injection vulnerability.
  - Use care when handing out personal, private, or secure data. Do not send this data to services you do not trust, and limit the number of services with which you share this data.

Problem 7

(40 pts.)

What is the difference between a GET and POST Request? Explain.

- A GET request only includes parameters included inside the URL (excluding cookies/browser data). GET requests parameters are appended to the URL using `?` and `&`. Example: http://www.purduecs.com/someaction.php?param1=hello%20world&param2=test
- A POST request includes additional key-value parameters in the request body, which are not included in the URL. POST requests can also contain additional multi-part data, used for handling file uploads and the like.
- Because GET requests included parameters in the URL, the request will be replayed if you visit the URL again (via a bookmark or history) as the parameters are included, while a POST request will need to be sent again entirely. Additionally, GET parameters will be logged wherever the URL is logged, such as on server logs, while POST parameters are not.

If you use HTTPS, will the query arguments of the GET be encrypted? Explain.

- Yes, because HTTPS uses Transportation Layer Security (TLS), all of the http request will be encrypted, including the GET query arguments. However, the destination server may still log the unencrypted URL/arguments it receives, which offers less security than POST.

If you use HTTPS, will the arguments of the POST be encrypted? Explain.

- Yes, because HTTPS uses Transportation Layer Security (TLS), all of the http request will be encrypted, including the POST arguments.

Why are GET requests considered less secure than POST Requests? Explain.

- GET requests are less secure because their arguments are included in the URL, and thus saved in web history, server logs, and more. Additionally, they are more easy to fake requests to.

Problem 8

List 5 things NMAP can do and the associated commands. If you are an attacker, how can you use NMAP as part of an attack? Explain. (70 pts.)

- Answer

Problem 9

List 5 things TCP Dump can do and the associated commands. If you are an attacker, how can you use TCP Dump as part of an attack? Explain. (70 pts.)

- Answer

Problem 10

From the readings on an Evening with Bernard and Stalking the Wiley Hacker explain how an attacker can conceal his/her location and identity when carrying out attacks. What challenges does this create for security professionals responsible for preventing these types of attacks (40 pts.)

- Answer

Problem 11

(60 pts.)

- What is the Raw Socket API?
  - Answer

- How can you use it to prevent/detect an attack?
  - Answer

- How can it be used to execute an attack?
  - Answer