CCD course - Description

- This used to be the Hacker's Hut course.
- Lecturers: Sandro Etalle, Luca Allodi, + a few guest lectures.
- NEW: LAB EXERCISE
- Students will also participate in a mandatory lab exercise (worth half a point at the exam) on threat detection. Participants will operate a real SOC for about 50 minutes, with the goal of detecting and reporting attacks that will be injected into the system by the lecturers. To gain the half point at the exam, participants must report correctly at least one of the injected attacks. The exercise will be run in groups of max 2 people; each group must have one laptop to participate in the exercise.
A suggested book

- It is not obligatory to use it, it is suggested.
Topic of this lecture

• This lecture: Quick Recap of web application security (part 1).
  • HTTP weaknesses, sessions, cookies. [Background Material: slides (45)]
• Next lecture (we might start today)
  • a. Recap of Web attack techniques: XSS, SQL injections, Path traversals, DDOS, Browser exploitation Frameworks. [Background material: slides (34)].
  • b. Structure of a targeted attack. Intelligence gathering, drive-by-exploit, watering hole, phishing, kill chain, living-off-the-lands. [Background material: slides (34)]
Disclaimer

- Any actions and or activities related to the material contained in these slides is solely your responsibility. The misuse of the information presented here can result in criminal charges brought against the persons in question.

- The author(s) and the TU/e do not promote, encourage and or endorse ANY illegal activities. The author(s) and the TU/e will not be held responsible in the event any criminal charges be brought against any individuals misusing such information to break the law.
You need to get your hands a bit dirty

• Please install
  • Something to tamper with the http request, such as liveHTTPHeaders or TamperData.
  • liveHTTPHeaders and TamperData do not work with the last version of Firefox. As alternative you can use Chrome with TamperChrome (a bit more complicated to use, but the result is the same, there is also a video tutorial).
  • 2018: What now works with my firefox is the add-on HTTP header live.
  • Install Firefox, look for a plug-in for “headers” and install it.
The trouble with webapplications

• Until a few years ago attacks Used to be on the Operating Systems

• Now it is easier to attack the (web) applications
  • Either for a direct attack (steal information)
  • Or an indirect attack (attack a user using the webapplication)

• Recent trend: attacking the OS/Silicon, again (e.g. Triton)

• Let’s take a look at OWASP
What do you see here?
What happens when you go to nu.nl?

• **Your browser (Internet Explorer, Firefox, Safari, Opera) will**
  • Display content coming from nu.nl
  • Display content coming from *external sites (tens of sometimes hundreds of them)*
    – So the browser will also transmit some information to these external sites
  • Display aggregate content from a number of advertisers

• Carry out some operations
  – Some of them determined by NU.nl
  – Some of them determined by some external sites
  – Input to these operations is partly determined by them
Simple Exercise

• Easy
  - Start using TamperChrome (or if you have an old version of Firefox TamperData) and take a look at all the http request that are originated by your browser when you go to www.nu.nl
  
  - login Facebook

• For the advanced user: install a proxy
  
  - Start burpproxy
  - under Kali: applications > web application analysis > burpsuite
  - make sure your browser is connected to the right proxy
  - typically: localhost:8081 (see the options in burpsuite)
This is a small fraction of the requests that are generated when you go to nu.nl.

Notice that they are mostly external sites.

...(this is using tamperdata and clicking on submit every time, tamperchrome and livehttpheaders have a different interface)
This is what it is

- Each box is a software component and it has vulnerabilities.
- And third party content is particularly untrusted.
Let’s play a bit
TamperData allows you to (and alter) an HTTP request …

In most cases the password is sent in clear (can be intercepted – unless https is uses)

- notice that you are able to modify the parameters

Obligatory exercise: activate “tamper data” and log in a site you know.
An HTTP request contains parameters

• The usual ones
  • Host
  • Referer. “The HTTP referer (originally a misspelling of referrer\(^1\)) is an HTTP header field that identifies the address of the webpage (i.e. the URI or IRI) that linked to the resource being requested. By checking the referer, the new webpage can see where the request originated.
  • Cookies

• The unusual ones
  • username,
  • password

• Important to understand: there exist two kind of parameters: GET and POST parameters.
Get parameter by example

- This is a page taken from the level 1 challenge on SQL injections.
- Type “ouch” in the “search” (Zoeken) field, press enter and you see....
Get parameter by example (2)

- That the parameter of the search is passed in the URL
- This is thus a GET method parameter
- Notice that you see something “different” in the URL also when you click on one of the buttons.
POST method by example (1)

• The old Facebook example,
• When you try to login you don’t see your login name and password in the url of facebook,
POST method by example (1)

- The parameters are now “hidden” in the request: they are thus POST parameters.
- You need to use temperdata to see them
- What are the consequences of this difference security-wise?
<table>
<thead>
<tr>
<th>GET</th>
<th>POST</th>
</tr>
</thead>
<tbody>
<tr>
<td>BACK button/Reload</td>
<td>Harmless</td>
</tr>
<tr>
<td>Bookmarked</td>
<td>Can be bookmarked</td>
</tr>
<tr>
<td>Cached</td>
<td>Can be cached</td>
</tr>
<tr>
<td>Encoding type</td>
<td>application/x-www-form-urlencoded</td>
</tr>
<tr>
<td>History</td>
<td>Parameters remain in browser history</td>
</tr>
<tr>
<td>Restrictions on data length</td>
<td>Yes, when sending data, the GET method adds the data to the URL; and the length of a URL is limited (maximum URL length is 2048 characters)</td>
</tr>
<tr>
<td>Restrictions on data type</td>
<td>Only ASCII characters allowed</td>
</tr>
<tr>
<td>Security</td>
<td><strong>GET is less secure compared to POST because data sent is part of the URL</strong></td>
</tr>
<tr>
<td>Visibility</td>
<td>Data is visible to everyone in the URL</td>
</tr>
</tbody>
</table>

- **GET parameter:** visible in the URL
  - You can bookmark it
  - Less secure: (e.g. when you only need to craft a link your victim has to click on)
- **POST parameter:** not visible in the URL
  - But sometimes the webapplication accepts transforming a post into a get parameter

Never use GET when sending passwords or other sensitive information!
... and an HTTP Response

<table>
<thead>
<tr>
<th>Response Header Name</th>
<th>Response Header Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td>OK - 200</td>
</tr>
<tr>
<td>Cache-Control</td>
<td>private, no-cache, no-store, must-revalidate</td>
</tr>
<tr>
<td>Expires</td>
<td>Sat, 01 Jan 2000 00:00:00 GMT</td>
</tr>
<tr>
<td>P3P</td>
<td>CP=&quot;Facebook does not have a P3P policy. Learn why here: <a href="http://fb.me/p3p">http://fb.me/p3p</a>&quot;</td>
</tr>
<tr>
<td>Pragma</td>
<td>no-cache</td>
</tr>
<tr>
<td>X-Content-Type-Options</td>
<td>nosniff</td>
</tr>
<tr>
<td>X-Frame-Options</td>
<td>DENY</td>
</tr>
<tr>
<td>Set-Cookie</td>
<td>datr=Md3hT0bgl-La0GG3BwWQVHFy; expires=Fri, 20-Jun-2014 14:54:05 GMT; path=/; domain=.facebook.com; httponlywd=...</td>
</tr>
<tr>
<td>Content-Encoding</td>
<td>gzip</td>
</tr>
<tr>
<td>Content-Type</td>
<td>text/html; charset=utf-8</td>
</tr>
<tr>
<td>X-FB-Debug</td>
<td>51HwMOf/1HVsaNhbd1Bfahcig+JwacTtsObqGGSP1Bc=</td>
</tr>
<tr>
<td>Date</td>
<td>Wed, 20 Jun 2012 14:54:05 GMT</td>
</tr>
<tr>
<td>Transfer-Encoding</td>
<td>chunked</td>
</tr>
<tr>
<td>Connection</td>
<td>keep-alive</td>
</tr>
</tbody>
</table>

The response contains interesting information:
- the status of the request.
- parameters that may be reused
Nb: some parameters are encoded

- This is to make the request more digestible by the webserver, webapplication etc.
- Very often requests are encoded,
- Very often the webapplication accepts several different encodings
  - Hexadecimal encoding: 0x41 = A, etc
  - URL encodings
    - %3d =
    - %25 %
    - %20 space
    - %0a new line
    - %00 null byte
One of the underlying problems: HTTP “weak” Sessions (1)

- **HTTP is connectionless**
  - In particular it is sessionless, and does not have a “state”
    - Small exception: HTTP keep-alive, or HTTP connection reuse in which you reuse the same TCP connection (Connection: Keep-Alive), this is however often unlinked to the application logic, so as far as we are concerned there is no connection state.
  - So, think about a web shop, in which you change “state” by putting things in the car etc.
  - The web application has no memory of what you have done so far.

- **Question: How can a server (dealing with 100s of connection at the same time)**
  - “know” which requests comes from you and which ones come from someone else?
  - Know the status of request, and what you have done earlier? Think about checking out at a webshop.
HTTP “weak” Sessions (2)

- All the information needed to process your request
  - Proof of who you are,
  - Information about what you have done so far (elements you have in the basket)
- IS STORED IN YOUR BROWSER
So, why don’t we use SSL and thereby “solve” all our problems?

• SSL DOES NOT HELP against the attacks we just mentioned!

• Actually, we should talk about TLS “Transport Layer Security”

• Where does SSL help?
Where SSL acts

- SSL takes care of securing the communication (TCP/IP).
- E.g. eavesdropping if you are at an internet café
  - Or at home using Wifi & WEP
  - Or WPA
- The security problems we mentioned before have nothing to do with the application vulnerabilities.
- Also: there are other ways of intercepting traffic.
If you have to develop any web application, you should look here for information how to avoid security problems

- Look at the cheat sheets.

Interesting for us is the OWASP top 10

- There is a tentative 2017 version
1. Injection
2. Broken Authentication
3. Sensitive Data Exposure [NEW]
4. XML External Entities (XXE) [NEW!]
5. Broken Access Control
6. Security Misconfiguration [Down from #5, still high enough]
7. Cross-Site Scripting (XSS) [Down from #3]
8. Insecure Deserialization [NEW!]
9. Using Components with Known Vulnerabilities
10. Insufficient Logging & Monitoring [NEW and interesting]

MANDATORY SELF-STUDY: THE 2017 OWASP TOP 10 (we might ask them)
The old OWASP

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 Control
A8: Cross Site Request Forgery (CSRF)
A9: Using Known Vulnerable Components
A10: Unvalidated Redirects and Forwards

credit to owasp.org
Cookies
Cookies in a nutshell (1)

- The HTTP answer (from the webapplication to the browser), may contain cookies that set a value in the browser
  - Eg. The response coming from amazon.com may contain
  - Set-Cookie: SessId=191041-1042
  - Set-Cookie: UID=1042
  - Set-Cookie: DiscountAgreed=25
- The next time we visit amazon.com the browser will include in the header the following information
  - Cookie: SessId=191041-1042
  - Cookie: UID=1042
  - Cookie: DiscountAgreed=25
- Nb: you can change this with temperdata
How cookies work

• Each cookie has a domain and a path
• It will be submitted to each request in which
  • The domain is a subdomain of the cookie domain
  • The path is a subdirectory of the cookie path
• Domain and path can be determined by the server in the so-called cookie policy
  • Set-Cookie: LSID=DQAAAK…; Domain=docs.foo.com; Path=/accounts
  • Provided that foo.com is the actual domain (or a subdomain) of the http request, otherwise the cookie will not be accepted for security reasons.
• Some cookies have an expiration moment
  • Some last only for the duration of the session
  • Some have an expiration date (e.g. June 15, 2025)
Third-party cookie and privacy

• What happens when you change these settings?
• Think about it a second.
Cookies and tracking

• We have seen that an HTTP request to nu.nl actually generates requests to other sites as well: google.com, amazon.com, advertiser.nl.

  • These sites (let’s call them collateral sites) will not be able to see the cookies set by nu.nl, but they will receive the cookies they have set in our browsers, even in previous sessions

• So what happens when we block “third party” cookies?

  • That these collateral sites will not be able to set new cookies in the browser, but they are still able to read their own (older) cookies. So tracking can take place anyhow.
Supercookies & IP addresses

• Supercookies are like cookies, but associated to a first-level domain: They are maintained by your internet provider
  • They are used also when the browser is in "private mode"
  • Are not eliminated when you clean the cookies from your browser
• Allow sites to track you even better.
• Tracking is also done via IP addresses:
  • I was looking at AIRBNB houses in Italy, and my son (upstairs) got some ads on the very same topic.
A very effective tracking method

Search

Default Search Engine
Choose the default search engine to use in the address bar and search bar.

- Google

- Provide search suggestions
- Show search suggestions in address bar results
A very effective tracking method

- Search suggestions
  - Don’t lie: it is probably “on” in your browser
- Exercise: turn on tamper data, and then type [www.yourfavoritesite.com](http://www.yourfavoritesite.com)
- What do you see?
- Each time you type a letter a new query goes to google, including all google’s cookies.
  - Google knows exactly what you type in the browser, all the time
  - And if you ever logged in google on that browser, google knows it’s you
Back to attacks (whether targeted or not)
Browser Exploitation Framework

• Take a look at beefproject.com
• Very nice video at
  • https://www.youtube.com/watch?time_continue=132&v=xdbvU_U42kY
• Features
  • Fingerprinting: Browser, OS, location fingerprinting
  • They adapt the response to the request (e.g. when they see a Firefox browser they put an exploit for Firefox in the response)
  • They can be used for targeted attacks (e.g. introduce the exploit only if browser originates from a given IP range)
Some BEPs

- Easy to find....

<table>
<thead>
<tr>
<th>Table 4.2 Most Widely Used BEPs List from Last 5 Years</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cool Exploit Kit</td>
</tr>
<tr>
<td>CritXPack</td>
</tr>
<tr>
<td>JustExploit</td>
</tr>
<tr>
<td>Hierarchy Exploit Pack</td>
</tr>
<tr>
<td>Lupit Exploit Pack</td>
</tr>
<tr>
<td>Nucsoft Exploit Pack</td>
</tr>
<tr>
<td>Sakura Exploit Pack</td>
</tr>
<tr>
<td>Salo Exploit Kit</td>
</tr>
<tr>
<td>T-Iframer</td>
</tr>
<tr>
<td>Zopack</td>
</tr>
<tr>
<td>Yang Pack</td>
</tr>
<tr>
<td>Yes Exploit</td>
</tr>
</tbody>
</table>
Watering Hole Attacks

- Gather Intelligence on the target
  - Discover suitable sites
- Infect steppingstone site
  - Possibly via 3rd party sites
- When target visits the site it is infected
  - Usually with a dropper
- Dropper on target contacts CC (command and control site) to download full malware
- Malware carries out the exploitation
  - Data exfiltration
  - Key logging
  - Etc etc etc
Phishing

• Email with malicious link/instructions
• Can be very targeted spear-phishing
• Like it appears coming from your boss
  • Sharing a document you should review
• In simple cases it redirects to a genuine-looking site where you need to fill some credentials
From: "MSU! Helpdesk" <macicchino@email wm.edu>
Date: January 30, 2014 at 10:41:52 AM EST
To: undisclosed-recipients:;
Subject: {Account Login Alert!}

This is an automated message to notify you that a valid password was used to login your MSU! account from an unrecognized device, Today Thursday, January 30th, 2014 at 09:00(UTC+02), in Mauritius, Port Louis (IP=41.136.181.172) as a result of that your account was temporarily suspended.

If you did this, you can safely disregard this email. If you didn't do this, kindly follow our review link below to retrieve your account
http://cse-msuaccountreviewauthenticationforum.yolasite.com/
Sincerely,
The MSU! Helpdesk
[---001:000564:57449---]
Please do not reply to this message. Mail sent to this address cannot be answered.
There is a whole market for DDoS attacks, you can just buy one. We’ll see some examples.