Things in security this week

Latitude breach

Shell vs reverse shell

- Reverse shell
    - If you send a reverse shell and send a payload for it to send something to you
    - Wait for a bit and you will run a command
    - Target sends things to the attacker
    - This works because the firewall doesn't check HTTP or HTTPS - the firewall won't check this because it looks like the target machine is sending get requests to a server

Shell - a way to get remote access to a computer
Reverse shell - a shell that goes the opposite direction
A situation

- When you access the target machine and you give a payload - you are asking the target to open a shell to wait for the next command
    - Then you send a command
    - Then it sends a response
    - Issue with this is if there is a firewall

What is a container

- E.g. docker
- Makes system calls down to the main operating systems
- Applications are all individual - runs applications in a sandbox environment and when it does need system calls
- Think about it as a bit of VM but a little less resource intensive because its sharing resources on the machine

If you can find that there is a vulnerability but you can't get the flag - put it in the report anyway just put up a writeup

MIDTERM FEEDBACK

- Writeups were mostly great

HTML REFRESH

- Html takes the form of <tag> data</tag>
- Scripts are common to get dynamic elements with JS used to manipulate objects

document.getElementsByTagName('body')[0].innerHTML = content

XSS

WHAT IS A SITE?

- Site is the composition of protocol, origin and port
- Origin is the entire domain, including subdomains and not including endpoints
    - An additional subdomain changes the origin and hence the site
- Http and https are different sites

Cross site scripting

- When you run some form of script that is unintended
- Is because you can reach beyond the bounds of a ‘site’
    - A malicious website could load another website into an adjacent frame or window then use JS to read into it
    - One website could cross a country and script into another page

TYPES

- Reflected
- Stored
- DOM
    - A mix between reflected and stored
- Self (kind of like reflected by not)

Reflected demo
Xss-game.appspot.com

To check for reflected xss - if you put something in the query or point where we get the same thing we put in the query reflected back to us

To run js we then go
<script>alert(1)</script>

- You have this when what you inject in the website is reflected by the application - returning what you write in rendered code of applications
- Good places to check
    - Search functions
    - Cookies
    - Strange client side variables like time
        - In the cookies or the request
        - Anytime that you type something and it gets rendered in the response you are looking at reflected XSS
- Reflected reflects back your response

Stored XSS Demo

- We write something and upon refresh its still there - stores it in the application
- If you send <script>alert(1)</script> and nothing puts up then try
- hello - if it is bolded then you know that it is loading JS
- Another way to load a script if script is not loading in our payload we can load it in an attribute
    - E.g.
    - <img src =”https://xss-game.appspot.com/static/level2_icon.png” onmouseover=”alert(1))”></img>
        - If you can’t get the image break and run this particular code
    - Now yes you run a script just requires some human interaction
    - Can also just go
      <img src =”x” onerror=”alert(1))”></img> - because this is something that will cause an error because its not an actual image so when it loads will just load up the error message
- Happens when the injected code is stored somewhere for future use
    - E.g. - comments, usernames, blog post titles and content body
- If filtering is not enabled then this exposes everyone to xss who views these things

Images are self closing so you don’t need a closing tag for those

MITIGATION TECHNIQUES

- Input filtration - good and useful but not foolproof
    - Some filtering works but nothing is ever foolproof
- **Don’t allow HTML to be input by the users**
- Response headers
    - Content type and x content type options - tells the browser to say don’t allow certain content types but this is up to the browser to implement
    - These aren’t always followed
- Use an up to date browser - Chrome 92 blocks “alert()” being called
    - Arbitrary JS for sandbox escape is bad
    - If you are not using an up to date browser that means there is a vulnerability in your browser
- 3 things to implement at the web app level
    - Same origin policy - SOP
        - Certain resources need to be loaded from the same origin
        - Is something that is already implemented for you - through firefox or chrome
        - This can be very restrictive so you need some way of sharing between origins - this is called CORS
    - CORS - cross origin resource sharing

        - Is allowed in the web browser response headers that say you are allowed to request resources from these particular sites
        - Is a form of relaxing the SOP
        - Browser needs a way to tell what is able to access a particular API
            - So CORS is set to say only Bank is allowed to provide JS to interact with the API for, in this example transferring payments
            - So here we depend on the browser to implement it correctly
            - Turning off CORS is very difficult - this happens in like the first 50ms of opening up or sending any request
        - XSS is all about tricking the user's browser into doing something that it is not supposed to do
        - API is the thing setting the rules here
    -

What can we do with XSS

- Steal a cookie and impersonate a user
- Carry out any action that the user can perform
- Capture login credentials
- Deface the website
- Mine cryptocurrency
- Anything because it runs javascript
    - Javascript allows you to arbitrarily interact with a site any way a person would

On chrome - you can't use alert() so just use print()

- They've just stopped the use of alert but other things work

POLYGLOT

- A PERSON THAT KNOWS A LOT OF LANGUAGES
- We can have exploits that take multiple forms
    - Polyglot scripts
    - If you had a jpeg and inside the jpeg it has a data section about metadata in that image
    - If you put alert(1) with script - sometimes this will be rendered with the photo

Anything someone can do with arbitrary JS they can do with XSS
If you click on some random website - can see how bad it can get because JS has essentially all access

- Be careful on what you click on as a result

Trust wave - place Jenson’s brother works