Think Like a Hacker:

Avoiding Common Web Vulnerabilities

KRISTINA BALAAM

Currently: Security Intelligence Engineer @ Lookout
Formerly: Application Security Engineer @ Shopify

@chmodxx
@chmodxx_
chmodxx
blog.chmodxx.net

SLIDES

http://github.com/chmodxx

OUTLINE

I. Why Do We Care?

II. A Crash Course In:

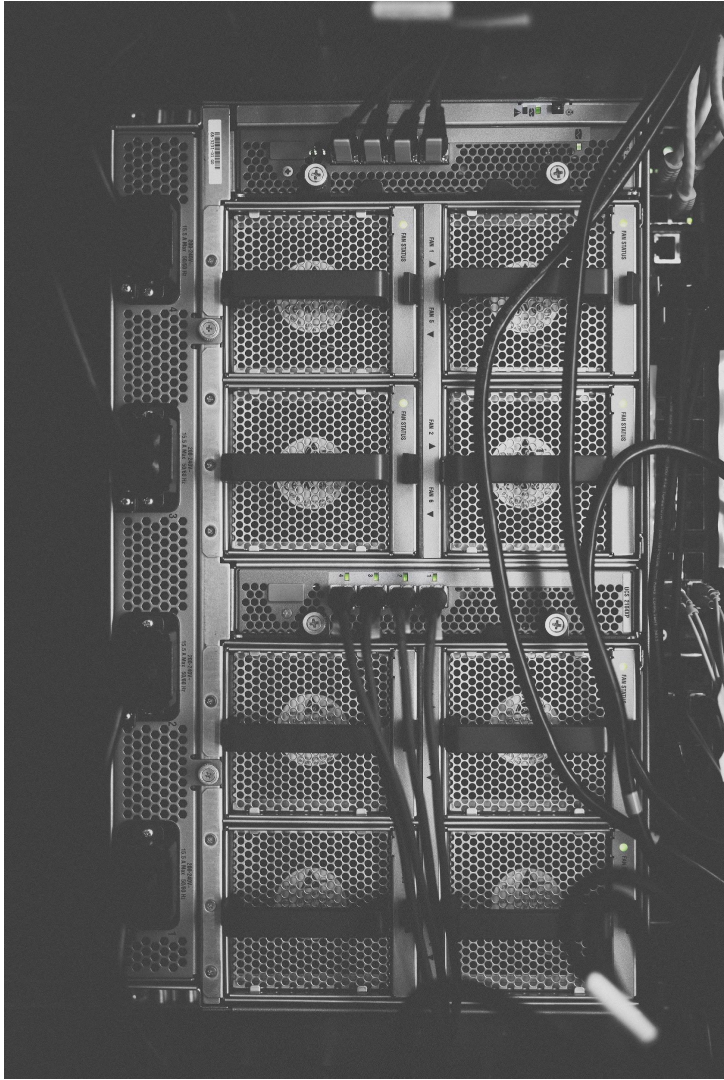
   A. Cross-Site Scripting (XSS)

   B. Client-State Manipulation

   C. Cross-Site Request Forgery (XSRF)

   D. SQL Injection

III. How to Keep Learning



BUT I’M NOT A DEVELOPER

https://www.identityforce.com/blog/2017-data-breaches

★ Most teams *don’t* have security engineers: for large tech companies, you’ll be lucky to have 1 AppSec Eng for every 100 developers

★ As a manager/owner, you set your agency’s culture and how you prioritize (or don’t) security

★ Security should be paramount to QA. Just because clients don’t see it, doesn’t mean it doesn’t matter.

Do I Really Need To Know This? *Yes*.

★ **Anyone can be targeted**. *Really*.

★ Schools, bootcamps don’t teach security (most of the time)

★ Invest in your dev’s education!

★ Targeted attacks can be devastating to clients *and* those responsible for not guarding against the attacks

CROSS-SITE SCRIPTING (XSS)

Basically code injection

People “inject” malicious scripts into websites that would otherwise be trusted

Usually client-side scripts

WHY IS THIS BAD?

Can lead to session cookies being compromised, which means a user’s session can be hijacked. Or, it can prompt an install-script for some sketchy program. Or, spoof some content.

E.G. from OWASP: “An XSS vulnerability on a pharmaceutical site could allow an attacker to modify dosage information resulting in an overdose”

FILE UPLOAD XSS

A file containing a malicious script is uploaded on a vulnerable site. Anyone who views that file can be affected.

E.g. Some sites that allow you to upload CVs, etc. and don’t restrict formats.

EXAMPLE

STORED XSS

Injected script is stored on the server somehow -- in a database, message forum, comment field.

“Victim” is presented the script when it requests information in some way.

EXAMPLE

REFLECTED XSS

Injected script isn’t stored on the server. The victim user has to open a link where the malicious script lives.

EXAMPLE

XSS IRL EXAMPLE

Stored XSS in developer.uber.com

“An attacker can make a series of requests to https://uber.readme.io/ that will result in permanent defacement/stored XSS of all the documentation pages on https://developer.uber.com/”

https://hackerone.com/reports/131450

HOW DO WE PREVENT THEM?

ESCAPE & SANITIZE ALL THE THINGS!

```
PHP: <input type="text" value="<?php echo esc_attr( $someValue );?>"/>

RoR: <%= h(potentially_unsafe_user_generated_content)%>
But as of Rails 3, this is handled automatically! Yay!
```

OWASP is handy:

XSS Cheat Sheet
https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

XSS Code Review Guidelines
https://www.owasp.org/index.php/Category:OWASP_Code_Review_Project

CLIENT-STATE MANIPULATION

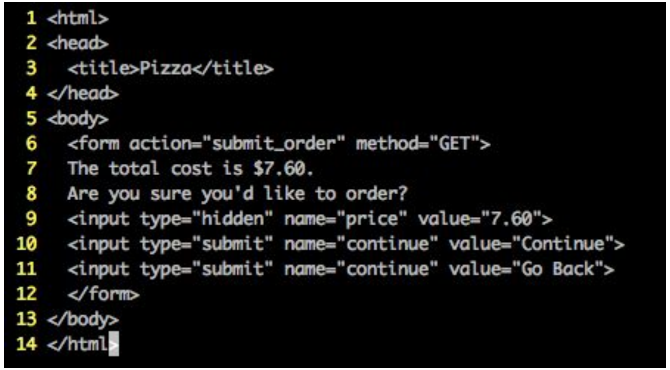
DO NOT TRUST THE CLIENT. NOT EVER!

EXAMPLE

Happens when servers provide state information to a client, and it's passed back as part of an HTTP request from the client.

EXAMPLES:

- Hidden input fields in a form when confirming a purchase
- GET instead of POST requests when submitting forms
- Privilege Escalation



```
<html>
<head>
  <title>Pizza</title>
</head>
<body>
  <form action="submit_order" method="GET">
  The total cost is $7.60.
  Are you sure you'd like to order?
  <input type="hidden" name="price" value="7.60">
  <input type="submit" name="continue" value="Continue">
  <input type="submit" name="continue" value="Go Back">
  </form>
</body>
</html>
```

HOW DO WE PREVENT IT?

NOT SENDING SENSITIVE INFO BACK TO THE CLIENT.

“SIGNED STATES” SENT BACK TO THE CLIENT.

Don’t make hidden data visible.

Don’t use GET for sensitive info.

Store a session-id and send *that* to the client rather than the explicit data value; that gets stored in the database where it’s used to look up the actual data.

Sign transactions - but make sure to sign *everything*.

CROSS-SITE REQUEST FORGERY (XSRF)

Attacker can use an HTTP request to modify user info from another site.

Servers aren’t really smart enough to determine whether an action was intentional; it just sees the request and does it.

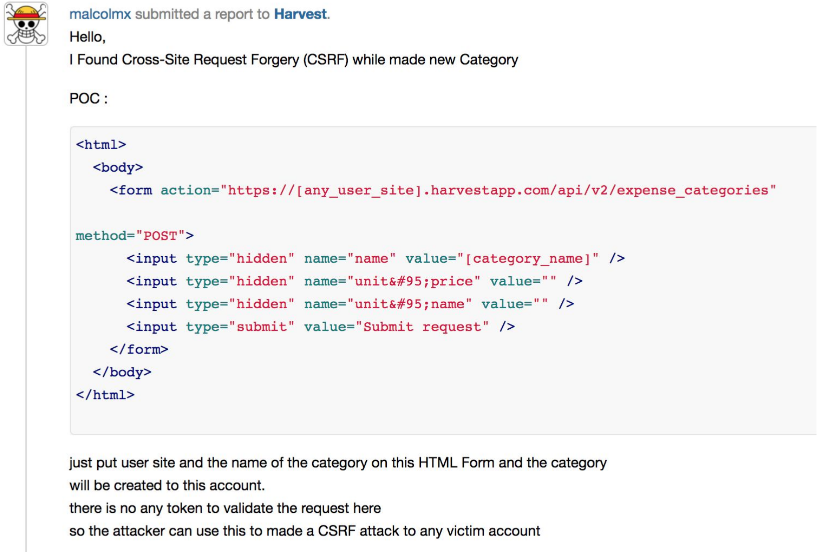
If a user has been authenticated on one site, and then is the victim of some kind of malicious script on *another* site: danger.

EXAMPLE

XSRF IRL EXAMPLE

Harvest, the time-tracking/invoicing app.

https://hackerone.com/reports/152569



malcolmx submitted a report to Harvest.

Hello,

I Found Cross-Site Request Forgery (CSRF) while made new Category

POC :

```
<html>
  <body>
    <form action="https://[any_user_site].harvestapp.com/api/v2/expense_categories"

method="POST">
      <input type="hidden" name="name" value="[category_name]" />
      <input type="hidden" name="unit&#95;price" value="" />
      <input type="hidden" name="unit&#95;name" value="" />
      <input type="submit" value="Submit request" />
    </form>
  </body>
</html>
```

just put user site and the name of the category on this HTML Form and the category
will be created to this account.
there is no any token to validate the request here
so the attacker can use this to made a CSRF attack to any victim account

HOW DO WE PREVENT IT?

TOKENS ARE YOUR BFF.

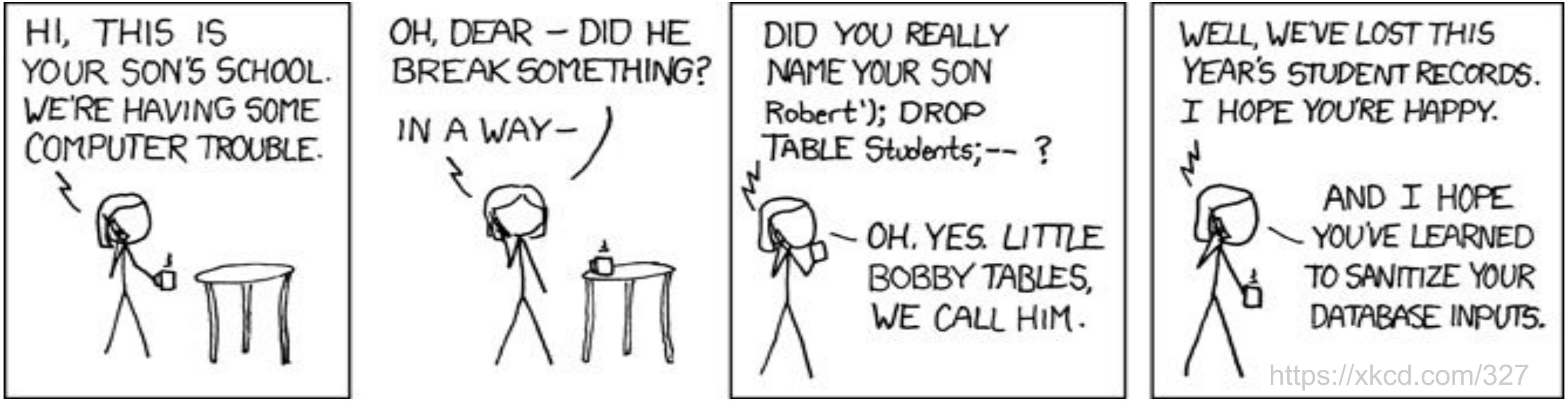
The site requires a special token before it’ll actually perform any data-altering operations. The token is submitted with the request, and validated by the “victim” site.

E.g. A bank

Validating origin headers submitted with the request

OWASP has a cheat-sheet for this too: https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)

SQL INJECTION


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

https://xkcd.com/327

AKA. “Command Injection Vulnerabilities”

End-user supplied input is inserted into a query or command to cause the command interpreter to misinterpret it.

EXAMPLE

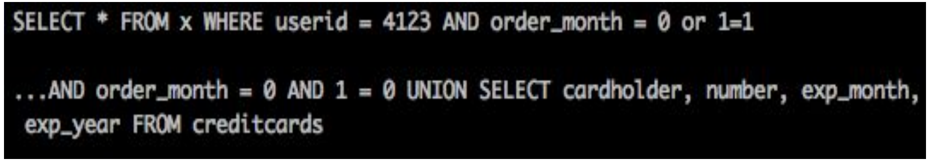
Sometimes “attackers” can give wonky input just to break a request and hopefully see error messages.

Can submit queries that complete a legit request and *then* start another:

E.g. … `month = 0; DROP TABLE x;`

Information leakage

E.g.



```
SELECT * FROM x WHERE userid = 4123 AND order_month = 0 or 1=1

...AND order_month = 0 AND 1 = 0 UNION SELECT cardholder, number, exp_month,
 exp_year FROM creditcards
```

SQL INJECTION IRL

Insurance product - LocalTapioca

“There exists an SQL Injection vulnerability on the path `/cs/Satellite`. This path accepts numerous fields, including `blobtable`, `blobcol`, `blobkey` and `blobwhere`. It is possible to sufficiently manipulate the the `blobwhere` field in order to trigger a time-based blind SQL Injection attack.”

https://hackerone.com/reports/164739

HOW DO WE PREVENT IT?

PREPARED STATEMENTS + BIND VARIABLES

Placeholders guaranteed to be data

Put placeholders where there exists something you should always consider data. E.g. `userid=?`

WHITELISTING

Specifying what input should look like and not allowing input that doesn't match pattern.

STUFF THAT DOESN'T WORK WELL:

Blacklisting doesn't work well. If you're looking for dangerous characters you could inadvertently conflict with functional requirements. (e.g. O'Brien)

Doesn't prevent numeric parameter attacks. Plus. attackers can use JS or encoding to specify characters that are blacklisted

Escaping doesn't guard against numeric params.

Stored procedures can still be used to perform malicious tasks by modifying the structure of the statement

HOW TO KEEP LEARNING

BOOKS

Peter Yaworski’s Web Hacking 101 eBook

The Web Application Hacker's Handbook

Foundations of Security: What Every Programmer Needs to Know

PODCASTS

OWASP Podcast

The Cyber Wire

Smashing Security

COURSES

Stanford’s Advanced Computer Security Certificate

Ryerson’s Digital Forensics & Cryptography Certificate

Coursera

PRACTICE, PRACTICE, PRACTICE

CTFs are fun, and good for teams!

Lots of websites for practicing:

- https://google-gruyere.appspot.com/
- ShellterLabs: https://shellterlabs.com/en/
- 22 Hacking Sites, CTFs & Wargames: https://wheresmykeyboard.com/2016/07/hacking-sites-ctfs-wargames-practice-hacking-skills/

THANK YOU!
Questions?