Oklahoma City
UNIVERSITY

Cybersecurity
CSCI 2413

Final Review

Protocols/Ports

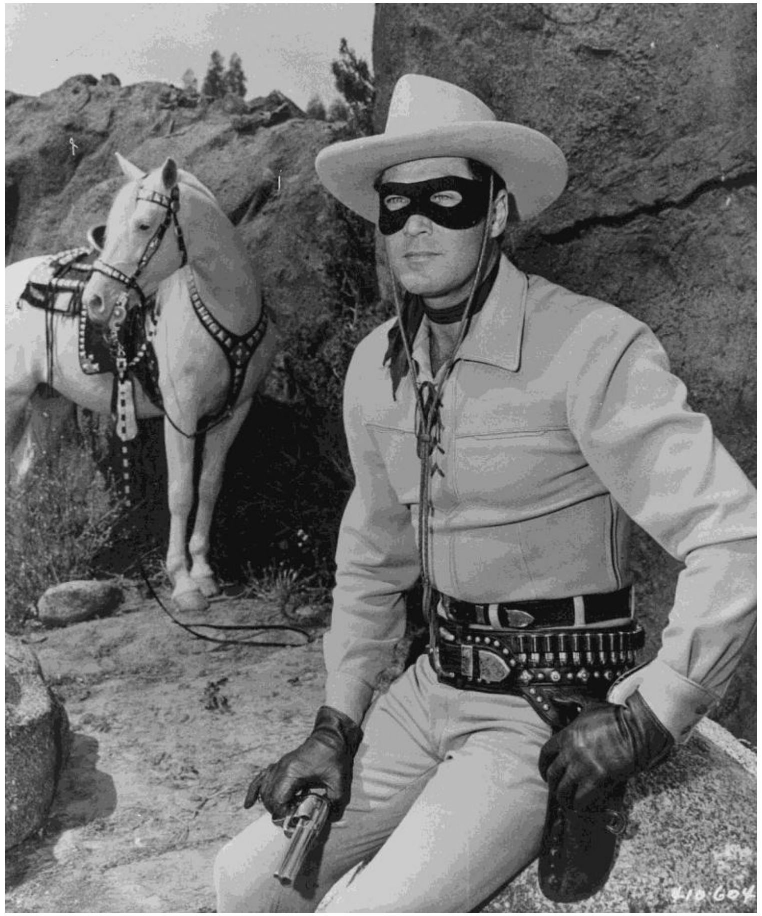
Protocol	Port
FTP	20, 21
SSH	22
Telnet	23
SMTP	25
DNS	53
HTTP	80
POP3	110
LDAP	389
HTTPS	443
RDP	3389

Types of Hackers

A ***white hat hacker*** is usually called a penetration tester today; It is someone who is hacking with permission of the owners of the target system

A ***black hat hacker*** is the person who gains access to a system in order to cause some type of harm; Black hat hackers are sometimes referred to as ***crackers***

A ***gray hat hacker*** is normally a law-abiding citizen, but in some cases will venture into illegal activities

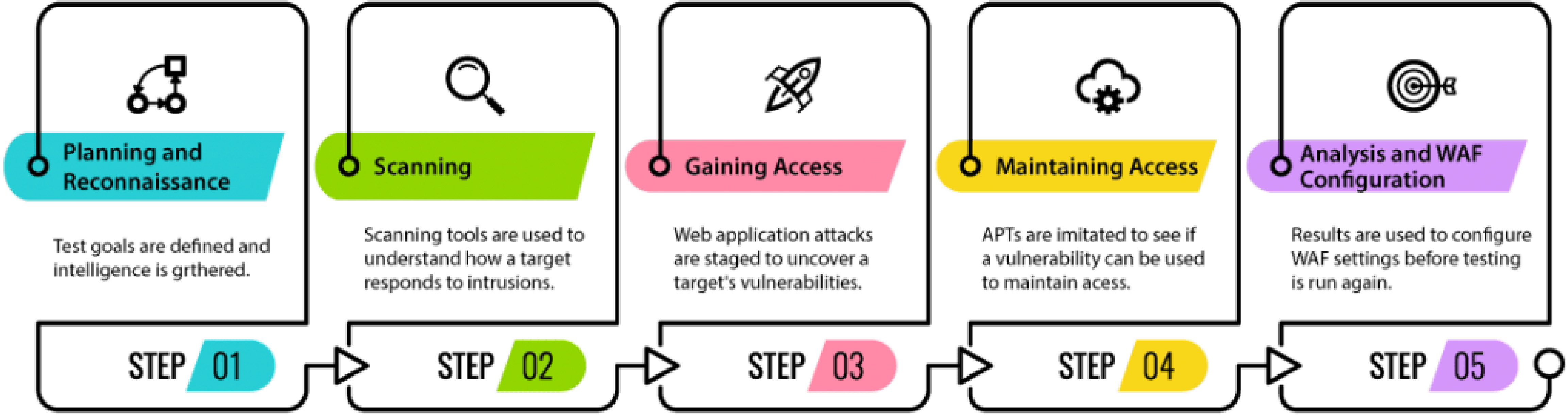


Penetration Testing (PenTester)

Also know as Ethical Hacker is some one that is **AUTHORIZED** to simulate a cyber attack against a computer or server.

Pen Testers are usually a company or security firm that performs several attacks on a company and provides a report to the company of any vulnerabilities found and how to fix any issues found.

Penetration Testing Stages



Planning and Reconnaissance

Test goals are defined and intelligence is grthered.

STEP 01

Scanning

Scanning tools are used to understand how a target responds to intrusions.

STEP 02

Gaining Access

Web application attacks are staged to uncover a target's vulnerabilities.

STEP 03

Maintaining Access

APTs are imitated to see if a vulnerability can be used to maintain acess.

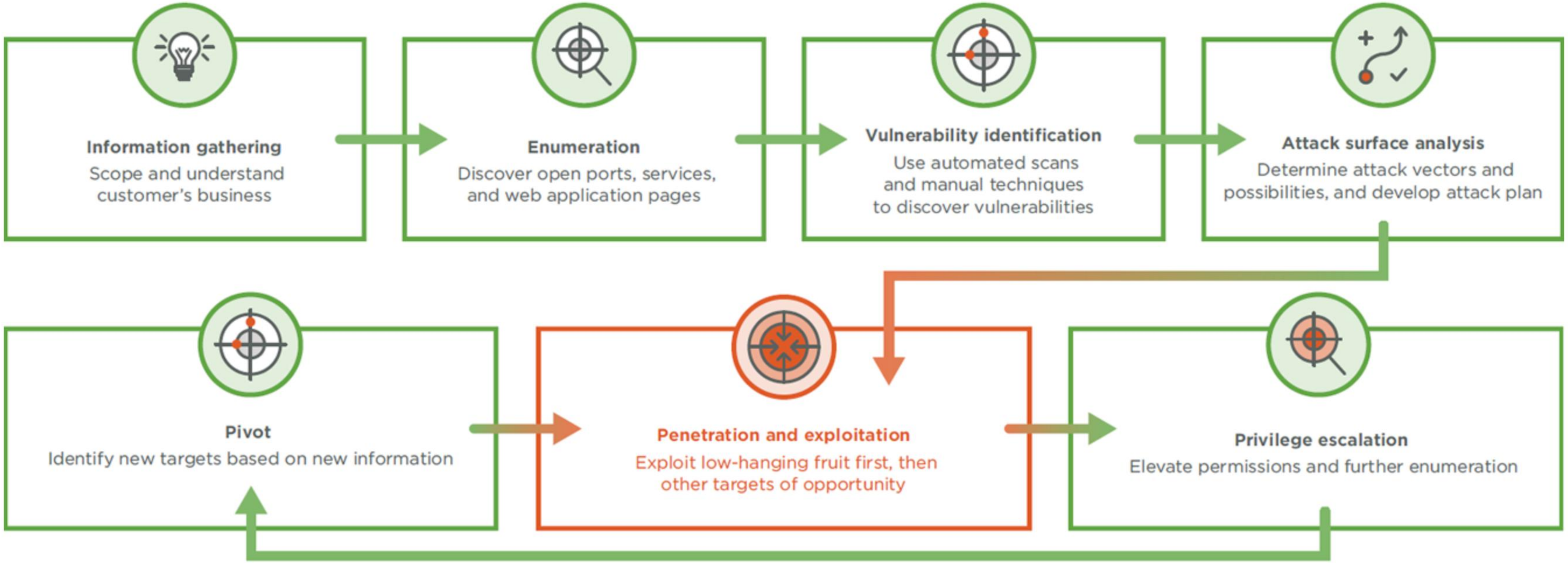
STEP 04

Analysis and WAF Configuration

Results are used to configure WAF settings before testing is run again.

STEP 05

Penetration Testing Stages



Information gathering

Scope and understand customer's business

Enumeration

Discover open ports, services, and web application pages

Vulnerability identification

Use automated scans and manual techniques to discover vulnerabilities

Attack surface analysis

Determine attack vectors and possibilities, and develop attack plan

Penetration and exploitation

Exploit low-hanging fruit first, then other targets of opportunity

Privilege escalation

Elevate permissions and further enumeration

Pivot

Identify new targets based on new information

OWASP

The Open Web Application Security Project (OWASP)

Non-profit foundation working to improve security of software

They Provide

- Tools
- Community and Networking
- Education and Training

https://owasp.org

OWASP Top Ten

OWASP releases a Top 10 most common flaws in software every 3-4 years

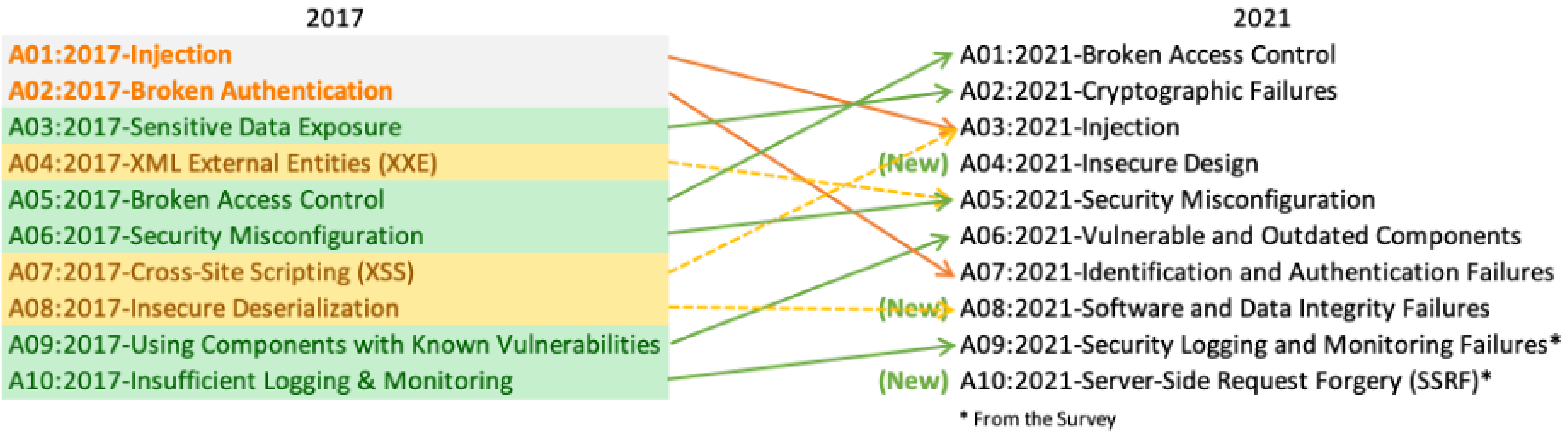
Current Top 10 is from 2021

Categories

- Exploitability
- Prevalence
- Detectability
- Technical

https://owasp.org/www-project-top-ten/

OWASP Top Ten (new in 2021)



2017	2021
A01:2017-Injection	A01:2021-Broken Access Control
A02:2017-Broken Authentication	A02:2021-Cryptographic Failures
A03:2017-Sensitive Data Exposure	A03:2021-Injection
A04:2017-XML External Entities (XXE)	(New) A04:2021-Insecure Design
A05:2017-Broken Access Control	A05:2021-Security Misconfiguration
A06:2017-Security Misconfiguration	A06:2021-Vulnerable and Outdated Components
A07:2017-Cross-Site Scripting (XSS)	A07:2021-Identification and Authentication Failures
A08:2017-Insecure Deserialization	(New) A08:2021-Software and Data Integrity Failures
A09:2017-Using Components with Known Vulnerabilities	A09:2021-Security Logging and Monitoring Failures*
A10:2017-Insufficient Logging & Monitoring	(New) A10:2021-Server-Side Request Forgery (SSRF)*

* From the Survey

https://owasp.org/www-project-top-ten/

A1: Broken Access Control

Access control enforces policy such that users cannot act outside of their intended permissions.

Failures typically lead to unauthorized information disclosure, modification, or destruction of all data or performing a business function outside the user’s limits.

https://owasp.org/Top10/A01_2021-Broken_Access_Control/

Vulnerabilities

- Violation of the principle of least privilege or deny by default.
- Bypassing access control checks by modifying the URL (parameter tampering)
- Permitting viewing or editing someone else’s account
- Accessing API with missing access controls for POST, PUT, DELETE
- Elevation of privilege acting as a user without being logged in
- Metadata manipulation, such as replaying or tampering with a JSON Web Token (JWT)
- Force browsing to authenticated pages as an unauthenticated user to privileged pages as a standard user.

Web HTTP Methods

The web sends data back and forth from server to the client using HTTP Methods

- GET – retrieves data from the server
- PUT – sends updated data to the server
- POST – sends new/created data to the server
- DELETE – removes data from the server

HTTP Status Codes

- There are several status codes that are returned from the server
  - 20X
    - 200 – OK
    - 201 – Created
    - 204 – No Content
  - 30X
    - 301 - Moved Permanently
    - 304 – Not Modified

- 40X
  - 400 – Bad Request
  - 401 – Unauthorized
  - 403 – Forbidden
  - 404 – Not Found
- 50X
  - 500 – Internal Server Error
  - 503 – Service Unavailable

How to Prevent

- Deny by default
- Use of Cross-Origin Resource Sharing (CORS)
- Disable web server directory listing
- Log access control failures and review periodically
- Make sure that the correct user is making the change
  Use tokens for authentication https://jwt.io
- Make Ids harder to guess (non-numeric)
  https://guidgenerator.com/

A2: Cryptographic Failures

Previously referred to as Sensitive Data Exposure

Protection of the data in transit and at rest.

- Passwords
- Credit Cards
- Health Records
- Personal Information (PII)

https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A3-Sensitive_Data_Exposure

Is the Application Vulnerable?

- Is any data transmitted in clear text?
- What version of cryptography is being used? SHA1, MD5….
- Are the default crypto keys in use weak? How were they generated?
- Is the certificate chain from the CA validated? Has it expired?
- Are the same keys being used on multiple servers?

Classify Data

Need to determine the sensitivity of all data elements being transmitted and stored

- Is it a company secret?
- Is there legal requirements for the data?
- Is it someone's Personally Identifiable Information (PII)?
- Is it someone’s Health Data (HIPPA)
- Is it a Password, PIN code, Security Question Answer?

Classify Data

Once the data has been classified determine the best way to store/transmit it.

- Passwords, PIN codes… - Hashed
- User personal information – Encrypted
- User health information - Encrypted
- Company Private Info - Encrypted
- Company Public info – Plain Text
- User Public info – Plain Text

Transmit Data

HTTP (Web) is not secure all web sites should use HTTPS (Secure Web)

FTP (File Transfer) is not secure all servers should use FTPS (Secure File Transfer)

SMTP (Mail) is not secure…. There is not a Secure Mail protocol but several mail programs/online solutions allow encrypting and decrypting of mail (Gmail)

Secure Email

There are add-ons for some email clients that allow you to encrypt and decrypt email messages it work on the Public Key encryption where both the sender and receiver have a public key they can use to encrypt messages and use their private key to decrypt.

Does take some coordination between both parties

Credit Cards

Payment Card Industry (PCI) have a set of standards to follow when storing or handling credit cards.

- They don’t encourage companies to store credit cards
- If you web site has reoccurring payments the recommendation is to use a token (encrypted value) from a credit card clearing company (First Data)

https://www.pcisecuritystandards.org/index.php

Data Storage

- Don’t store sensitive data if you don’t have to.
- Truncate the data (Last 4 of SSN)
- At least encrypt it if you must store it
- Move the data to an internal system if your site doesn’t need the data
- Hash data that just needs to be compared/matched and not displayed.

How to Prevent?

- Classify data processed, stored, or transmitted into sensitivity levels
- Classification my have to take in consideration laws and reguations
- Don’t store sensitive data if you don’t have to
- Encrypt all sensitive data at rest
- Use up to date and standard cryptography
- Use TLS 1.2 or above
- Disable caching sensitive data

A03: Injection Attacks

Injection is most commonly exploited through a SQL Injection attack but there are many injection attacks including:

- SQL, NoSQL, Command, OS and LDAP

Injection attacks result from untrusted data being sent or processed without validation

Is the Application Vulnerable?

User-supplied data that is NOT validated

Dynamic queries are not parameterized

Hostile data is directly used or concatenated in the input

Relational Databases

One of the most efficient ways to store data is in a relational database

Relational databases have Tables and those tables have columns and rows similar to an Excel spreadsheet

These tables can be queried using a language called Structured Query Language (SQL)

Relational Database Options

FREE

MySQL

POSTGRES

Enterprise

Microsoft SQL Server

Oracle

IBM DB2

MySQL

https://www.mysql.com/

Owned by Oracle

One of the most popular FREE Relational Databases

Provides Server and management consoles for multiple operating systems (Windows, MacOS, Unix)

Tables, Rows, Columns

Every table is given a name like “Users”, “Products”…

Tables have columns of like “Name”, “Username”, “Email”…

Tables can have multiple rows of data

Id	Name	Username	Email	Password
1	Jeff	Jmaxwell	jmaxwell@okcu.edu	Password#123
2	John	Jsmith	jsmith@okcu.edu	Password#123
3	Tom	Hanks	thanks@okcu.edu	Password#123

Create, Read, Update, Delete (CRUD)

Relational database provide a way to **C**reate data, **R**ead data, **U**pdate data, and **D**elete data (CRUD)

In SQL we use the following keywords:

- CREATE = INSERT INTO
- READ = SELECT
- UPDATE = UPDATE
- DELETE = DELETE

READ - SELECT

SELECT [columns] **FROM** [table]

[columns] can be one or multiple columns in a comma delimited list or “*” which means all columns.

Examples:

SELECT * **FROM** Users

SELECT Name, Email **FROM** Users

WHERE Clause

WHERE Clause allows filtering rows in the table by specific columns

SELECT [columns] **FROM** [table]
WHERE [column] = [value]

Examples:

SELECT * **FROM** Users **WHERE** Name='Jeff'

Id	Name	Username	Email	Password
1	Jeff	Jmaxwell	jmaxwell@okcu.edu	Password#123

WHERE Examples

SELECT * **FROM** Users **WHERE** Id=2

Id	Name	Username	Email	Password
2	John	Jsmith	jsmith@okcu.edu	Password#123

SELECT * **FROM** Users **WHERE** Name **LIKE** ‘J%’

Id	Name	Username	Email	Password
1	Jeff	Jmaxwell	jmaxwell@okcu.edu	Password#123
2	John	Jsmith	jsmith@okcu.edu	Password#123

CREATE – INSERT INTO

INSERT INTO [table] ([columns]) **VALUES** ([values])

When inserting (creating) data in a table a comma delimited list of columns and values are passed.

Examples:

INSERT INTO Users (Id, Name, Email,….) **VALUES** (4, ‘Hank’, ‘hank@okcu.edu’,….)

UPDATE – UPDATE

UPDATE [table] **SET** [column]=[value]

WHERE [column]=[value]

Need to be careful when updating data and include a WHERE clause otherwise all records in table will be changed.

Examples:

UPDATE Users **SET** Name='Jeff Maxwell'

WHERE Id=1

DELETE – DELETE

DELETE FROM [table]

WHERE [column]=[value]

Need to be EXTRA careful when deleting data and include a WHERE clause otherwise all records in table will be DELETED.

Examples:

DELETE FROM Users **WHERE** Id=1

SQL Query

SELECT * FROM Users

Id	Name	Username	Email	Password
1	Jeff	Jmaxwell	jmaxwell@okcu.edu	Password#123
2	John	Jsmith	jsmith@okcu.edu	Password#123
3	Jane	jsmith2	jsmith2@okcu.edu	Password#123

How SQL Injection Works

A login screen requires a username and password, which is checked against a database

SQL injection adds OR X=X to the end of the password

- The result of X=X is always true, so that string will return a value of True to the website
- If the code is not well written, this can trick the website into believing the correct password has been entered

SQL Injection

SELECT * FROM Users

WHERE username=‘jmaxwell’ AND password=‘Password#123’;

```
String sql = “SELECT * FROM Users WHERE
username=‘“ + username + “’ AND password=‘” +
password + “’”;
```

SQL Injection

What if pass is jmaxwell' or '1'='1' --

```
String sql = “SELECT * FROM Users WHERE
username=‘“ + username + “’ AND password=‘” +
password + “’”;
```

```
SELECT * FROM Users WHERE username=? AND
password=?
```

SELECT * FROM Users

WHERE username=‘jmaxwell’ **or ‘1’=‘1**’ **--** AND password=‘xxx’

A04: Insecure Design (New in 2021)

Insecure Design is a very broad category including a lot of potential issues:

- "Missing or Ineffective Control Design"
- What are the Risks?
- How are the Risks being managed?

What is Secure Design

Secure Design is not just a diagram it is more of a change in thinking, culture and approach to building software and hardware systems.

You need methodologies and a constant drive as a corporation to make things more secure starting with the Design.

What is Secure Design

Secure Design is not just a diagram it is more of a change in thinking, culture and approach to building software and hardware systems.

You need methodologies and a constant drive as a corporation to make things more secure starting with the Design.

Threat Modeling

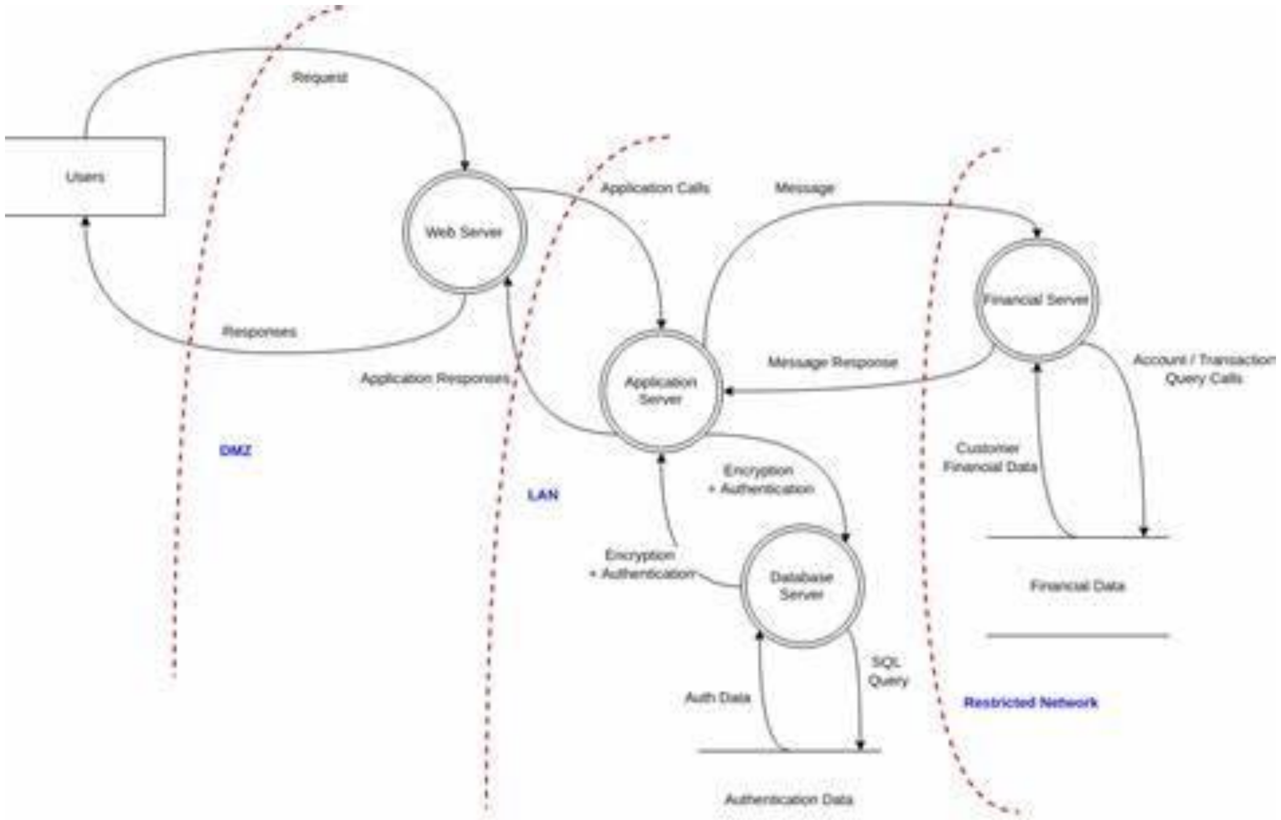
Threat Modeling is the practice of reviewing current Application diagrams and look for potential secure issues.

How does the data flow?

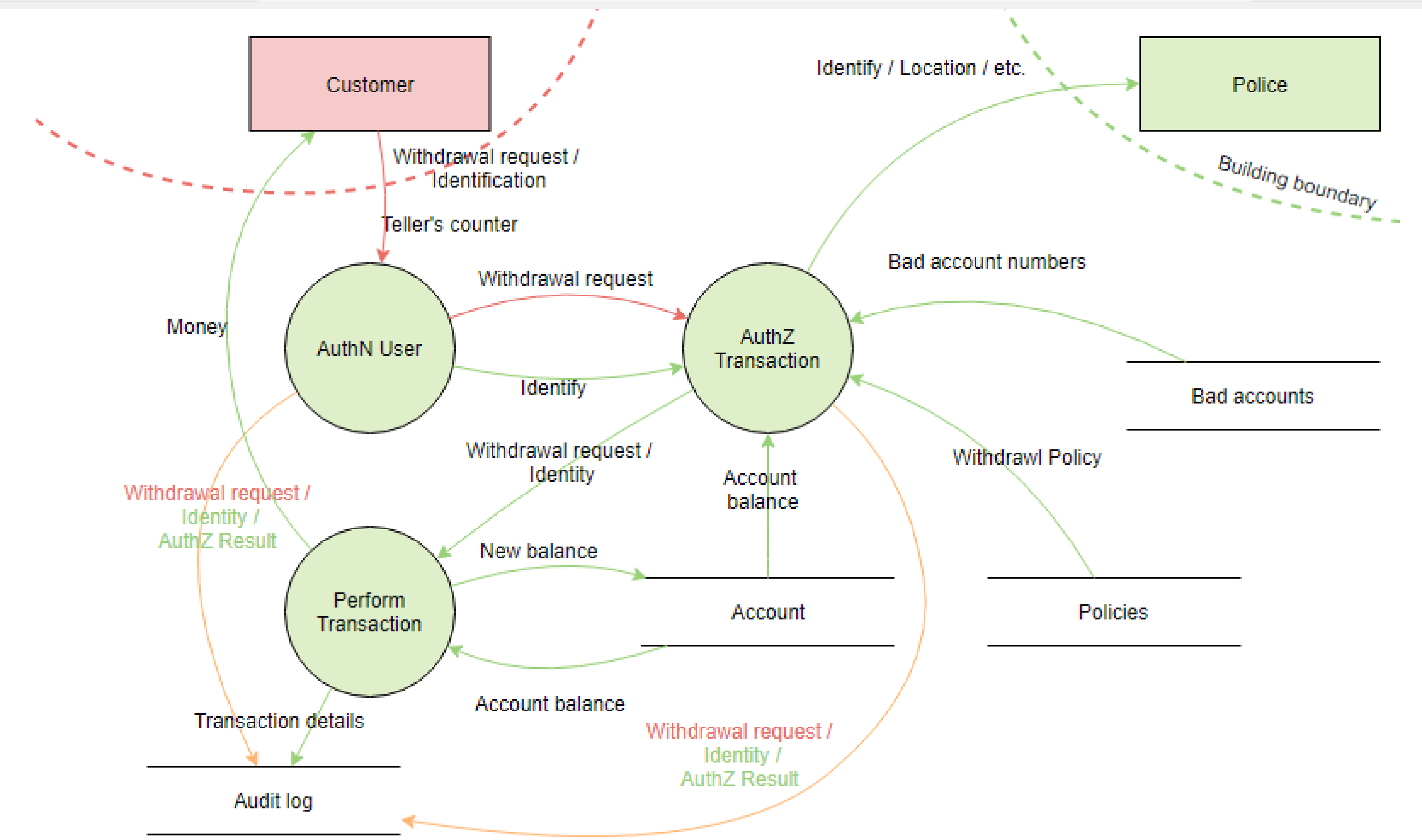
Is Encryption being used?

Who has access to WHAT?

Threat Modeling



Threat Modeling



Data

- What data is being stored?
- How is the data classified?
- What data is being Hashed/Encrypted? Why or Why Not?
- Who has access to the data?
- What data is being sent to other sources (outside companies)?

Data Flow

- How does the data flow between systems?
- What are the inputs/outputs?
- Is the data transferred in plain text or cipher text?
  - What encryption methods are being used for cipher text?

Security Assessment

- Has a security Assessment been performed?
  - When was the last one?
  - What were the findings?
  - How was it performed?

- OWASP Software Assurance Maturity Model (SAMM) https://owaspsamm.org

How to Prevent

- AppSec (Application Security) professionals need to be a part of the design and discuss security early in the design progress
- Build a library of Secure Design patterns or reference example code that is secure for the teams to use.
- Use Threat Modeling for critical systems and diagram the authentication, access control and authorization patterns
- Diagram/Document the Data flows between systems

How to Prevent

- Create User Stories/Requirements to test and validate that your system is not vulnerable to known attacks (SQL Injection, XSS, CSRF…)
- Test and validate your design
- Have a PenTester try to hack the application/servers in the system

A05: Security Misconfiguration

Security Misconfiguration deals with systems and software not being setup properly.

Attackers like to target unpatched flaws that other attackers have already found.

Attackers look for unprotected files and folders

A lot of default configurations leave servers vulnerable

A05: Security Misconfiguration

- Lack of OS/System Hardening
- Missing latest patches or security patches
- Services running that shouldn’t be
- Features installed that need to be removed
- Open Ports or Protocols
- Error handling that shows stack trace with security information
- Default settings sometimes allow too much access

Why does this happen?

- A lot of administrators setup systems and don’t know what default values are configured
- Companies do not use automated tools to patch their systems and sometimes systems are missed
- Default configurations are used

How to Prevent

- Work with administrators and network admins to “Harden” the server “Lock it down”
- Remove any default accounts
- Remove any unused Services
- Turn off any unused Ports
- Add default error handling
- Take a least privilege approach to everything

How to Prevent

- Create a secure version of each OS that can be used when building new systems
- Security Testing and automated tools

- Log Server Errors and Applications errors (AND LOOK AT THEM)

A06: Vulnerable and Outdated Components

- In modern development we use a combination of 3rd party components to provide us with features/functionality.

- A lot of these components could have a vulnerablity

Is my application vulnerable?

- Probably
- If you use any 3rd party components
- If you are not regularly scanning for vulnerabilities
- If you are several versions behind on the framework or tools you use

Why does this happen?

- A lot of 3rd party software is Open Source (hosted on GitHub or GitLab) and it is up to the author to update their software, scan for vulnerabilities, fix and deploy new versions.
- If my Open Source Project doesn’t make any money it might not be a high priority for me.
- All commercial (paid for) software should patch their software regularly (if not look for new software)

Tools

- National Vulnerability Database (NVD)
  https://nvd.nist.gov

- GitHub/GitLab have built-in scanners

- OWASP Dependency Checker
- https://owasp.org/www-project-dependency-check/

- Snyk.io
  https://snyk.io

NPM

- Node Package Manager (NPM)
- `npm ls [package name]`
- `npm audit [fix]`
- `npm update`

How to Prevent

- Patch, Patch, Patch as a normal process in the development cycle
- Remove Unused Dependencies (This can be challenging)
- Use a scanner or add a scanner into your build/deployment pipelines (CI/CD)
- Only use components from an official site/source

How to Prevent

- Create a local repository that the development team uses to pull updates and update the repository regularly
- Have scheduled releases updates (once a quarter, month, or week)
- Don’t let the frameworks (in your application) get behind more than one or two versions.

A07: Identification and Authentication Failures

- Formally #2 (in 2017) and named “Broken Authentication” moves to #7 due to increased secure and better management of Authenticated users.

- **Authentication** relates to session management or logins that are implemented incorrectly.
- Hackers can exploit the and login as other users or gain access to pages or systems they should not have access to.

Does your Application?

- Permit automated attacks such as credential stuffing, where the attacker has a list of valid usernames and passwords?
- Permits brute force or other automated attacks?
- Permits default, weak or well-known passwords, such as “Password1” or “admin/admin”?
- Uses weak forgot-password processes – knowledge based answers

How is login validated/manage?

- Uses plain text, encrypted or hashed passwords without salt (A02: 2021 Cryptographic Failures)
- Not using Multi-factor Authentication
- Pages that don’t validate the user session or validate access to a page

Credential Stuffing

The hacker uses stolen account credentials form other hacks and tries them on other sites.

Have I Been Pwned? Has over 9 Billion hacked accounts from previous breaches.

Adobe
EQUIFAX
LinkedIn
YAHOO!
ebay
MARRIOTT
Heartland
Payment Systems

https://haveibeenpwned.com/

Brute Force Attacks/Weak Passwords

Taking commonly used passwords and try them for each account

https://github.com/danielmiessler/SecLists/blob/master/Passwords/darkweb2017-top10000.txt

Easy to exploit if system allows simple passwords.

HASHCAT

Weak forgot-password process

Have you ever used a forgot password feature and it emailed you your password?

- NOT HASHED

What if you enter you email for a forgot password and it says that email/username does not exist?

- Why is that bad?

Plain text passwords or Encrypted Passwords

Plain text ARE BAD

- You should hopefully KNOW this by now.

Encrypted passwords are a little better but anything encrypted can be decrypted so it is also BAD.

You should ALWAYS Hash passwords with a RANDOM Salt value for each user.

How to fix Broken Authentication?

Implement Complex Password Requirements

Implement Account lockout with too many failed attempts****

HASH passwords with a random SALT

Implement Multi-Factor Authentication

Credential Stuffing

Keep track of the number of failed login attempts and alert on a threshold to the user/admins

Implement Multi-Factor (Email, Text, QR code scan, RSA Token)

https://haveibeenpwned.com/

Brute Force Attacks/Weak Passwords

Keep a bad password list and when a user signs up or tries to change their password check to see if it is NOT in the bad password list.

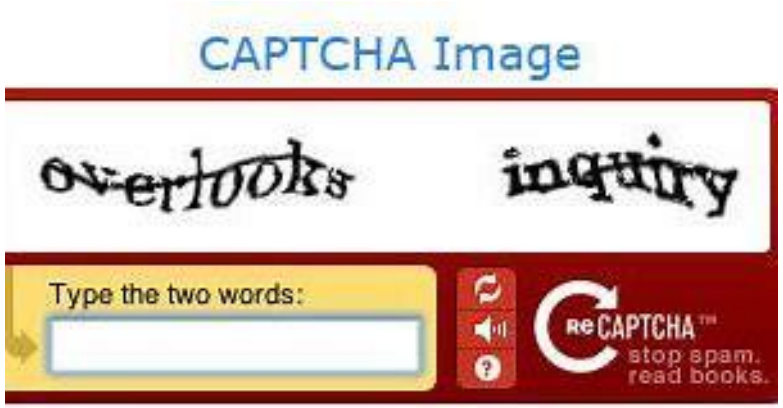
Setup complex password rules

- Passwords must be 8 or more characters (> 15 is better)
- Passwords must contain both upper and lower case letters
- Passwords must contain at least 1 special character (!@#$%^&*)

Brute Force Attacks/Weak Passwords

Don’t have a small limit of how long a password can be

- “Hello my name is Jeff Maxwell and I like to Teach #1 3 5” – Is harder to hack than
  P@$$w0rd123



ComputerHope.com

Weak forgot-password process

Never email the user their password (EMAIL is in clear text across the web)

Don’t tell the user (hacker) their email/username doesn’t exist. If you do then can use the forgot password to narrow down the list of users who have accounts before they attack

Instead always say something like “If an email/username exists password reset instructions will be emailed” (continue)

Weak forgot-password process

Instead always say something like “If an email/username exists password reset instructions will be emailed” (continue)

If someone requests to reset their password send them a link (URL) with a random code that is associated with the user account if they go to the link let them reset their password

Ask them to type in a code from a text.

Plain text passwords or Encrypted Passwords

You should **ALWAYS** Hash passwords with a **RANDOM** Salt value for each user.

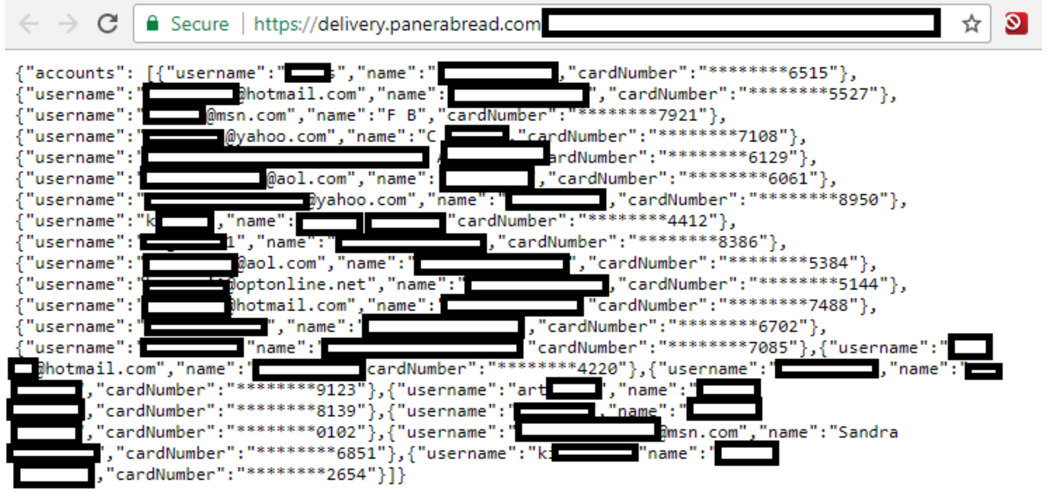
Session Management

- Once a user has logged in you need to make sure that each page they access they should have access to that page.

Session Management

- Panera Bread was notified that they were not validating users when they navigated between pages and wasn’t using the correct Session Management restricting access.

- https://panerabread.com/users/1234 could be changed to access other user accounts



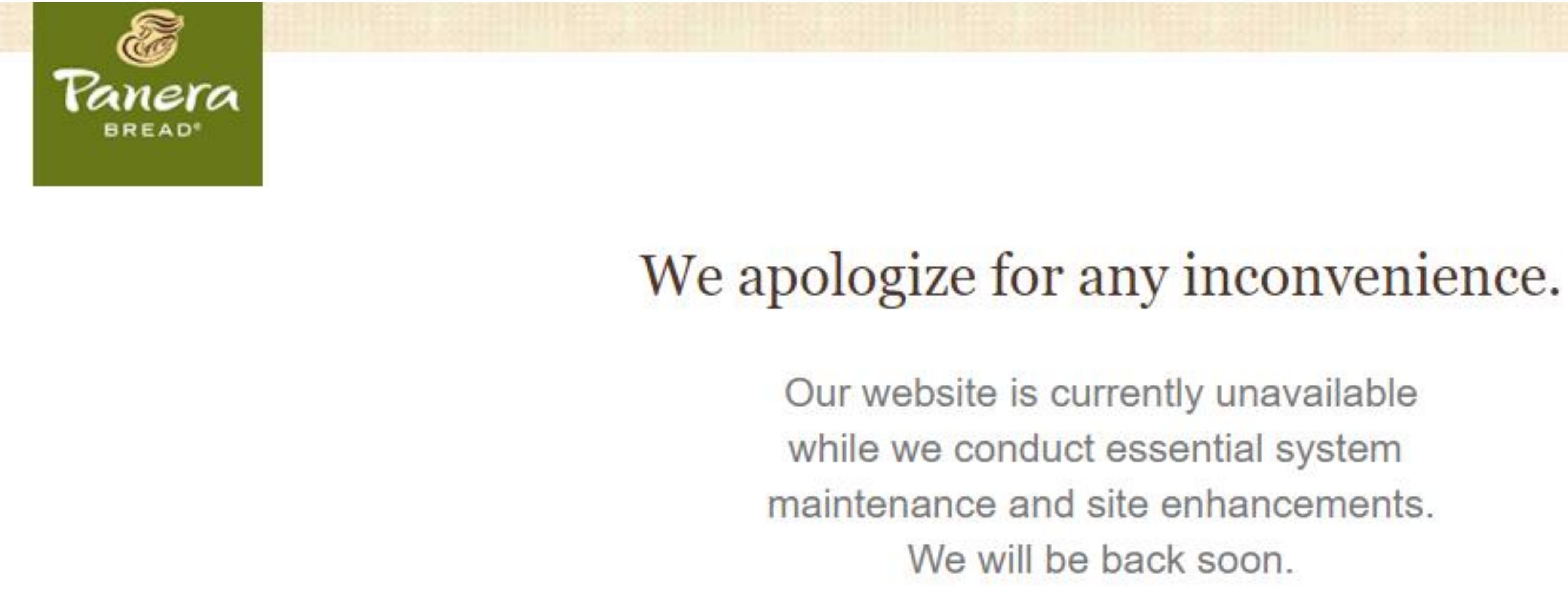
Secure | https://delivery.panerabread.com[redacted]

```
{"accounts": [{"username":"[redacted]","name":"[redacted],"cardNumber":"********6515"},
{"username":"[redacted]@hotmail.com","name":[redacted]","cardNumber":"********5527"},
{"username":"[redacted]@msn.com","name":"F B","cardNumber":"********7921"},
{"username":"[redacted]@yahoo.com","name":"C [redacted]"cardNumber":"********7108"},
{"username":"[redacted] [redacted]rdNumber":"********6129"},
{"username":"[redacted]@aol.com","name":[redacted],"cardNumber":"********6061"},
{"username":"[redacted]@yahoo.com","name":"[redacted],"cardNumber":"********8950"},
{"username":"k[redacted],"name":[redacted] [redacted]"cardNumber":"********4412"},
{"username":[redacted]1","name":"[redacted],"cardNumber":"********8386"},
{"username":"[redacted]@aol.com","name":"[redacted]","cardNumber":"********5384"},
{"username":"[redacted]@optonline.net","name":"[redacted],"cardNumber":"********5144"},
{"username":"[redacted]@hotmail.com","name":[redacted]"cardNumber":"********7488"},
{"username":"[redacted]","name":"[redacted],"cardNumber":"********6702"},
{"username":"[redacted]"name":"[redacted]"cardNumber":"********7085"},{"username":"[redacted]
[redacted]@hotmail.com","name":"[redacted]cardNumber":"********4220"},{"username":"[redacted],"name":"[redacted]
[redacted],"cardNumber":"********9123"},{"username":"art[redacted]","name":"[redacted]
[redacted],"cardNumber":"********8139"},{"username":"[redacted],"name":"[redacted]
[redacted],"cardNumber":"********0102"},{"username":"[redacted]@msn.com","name":"Sandra
[redacted]","cardNumber":"********6851"},{"username":"k[redacted]"name":"[redacted]
[redacted],"cardNumber":"********2654"}]}
```

Panera Bread – KrebsOnSecurity

- Dylan Houlihan reached out to Panera Bread and they said they were aware and were working on it.
- 8 months later Dylan had not heard anything and the problem still existed so he reached out to Brian Krebs at https://krebsonsecurity.com
- Krebs reached out to them with no response published a blog post about it https://krebsonsecurity.com/2018/04/panerabread-com-leaks-millions-of-customer-records/ and several newspapers/news channels picked up the story.

Panera Bread – KrebsOnSecurity



We apologize for any inconvenience.

Our website is currently unavailable while we conduct essential system maintenance and site enhancements. We will be back soon.

A08: Software and Data Integrity Failures

- This is a new category added for 2021 and deals with making the assumptions that things are working without verification

Software and Data Integrity

- Relates to code and infrastructure that does not protect against integrity violations
- Cloud Servers, Content Delivery Networks (CDNs) and insecure CI/CD pipelines for deployment
- Many applications now include auto-update functionality where updates are downloaded without sufficient integrity verification

How to Prevent

- Digital Signatures to validate who created the update
- Manage your own internal NPM or NuGet libraries for 3rd party controls
- After each deployment calculate a HASH on the files/folders and compare the the HASH that is expected (run this at least once a day)

How to Prevent

- Security and Code Scans
- Manual Code Reviews of changes
- CI/CD don’t store passwords and limit access

A09: Security Monitoring and Logging Failures

Oklahoma City
UNIVERSITY

- Previously known as “Insufficient Logging and Monitoring” this item moves up from #10 to #9

- This mainly deals with traffic and the health of an application

Logging

- Capture all traffic and activity to a site
- Disk Space is cheap log as much as you can offload to another server
- Never leave the logs on the server in case the server is compromised and altered by the hacker

Logging

- What is the application logging by default?
- What should you be logging?
- What should you NOT log?
- Are the logs stored locally?
- Are there any errors?

What should you NOT log?

- Any Personally Identifiable Information (PII)
- Credit Cards, Bank Numbers, ….
- Tokens, Security Keys….
- Passwords

Types of Logging

- What kind of logging does the code have?
- Exception Logs – when the application/code crashes unexpectedly
- Audit Logs – Logging every Page, Event, Transaction, ….
- Usage Logs – How many times a Page is viewed or a feature is used?
- Security Logs – Login Success and Failure, Access Denied

Logging Rules

- Log normal traffic (successful paths)
- Log any errors
- Find a tool to manage the Logs
  - Splunk

Security Logging

- Log Logins:
  - Successful Logins
  - Failed Logins
- All warnings and errors should be logged
- Log abnormal traffic

Monitoring

- Logging is **WORTHLESS** unless you review the logs
- Need to review any errors in the system and report them to the development teams daily
- Need a system to alert on certain thresholds or when certain types of traffic occur

How to Prevent

- Ensure all login, access control, and server-side input validation failures are logged and flagged as potential hack attempts
- Ensure that logs are generated in a format that is easily consumed by logging tools
- Make sure logs are encoded correctly so they are not vulnerably to SQL Injection or XSS

How to Prevent

- Ensure high-value transactions have a detailed audit trail with integrity controls to prevent tampering or detection such as append-only database tables or blockchain

- Alerting for suspicious activity should be established and reports/dashboards with he team should be visible to everyone

How to Prevent

- Create an incident response and recovery plan teams to respond/react if a hack occurs

A10: Server-Side Request Forgery (SSRF)

- This was previously on the OWASP top 10 but dropped off in the 2017 list

- But it is BACK!!!

A10: Server-Side Request Forgery (SSRF)

- SSRF occurs whenever a web application is fetching a remote resource without validating the user-supplied URL.

- It allows attackers send well crafted requests to return unexpected results or access sites that the user shouldn’t have access

A10: Server-Side Request Forgery (SSRF)

- https://site.com/users/1234

- If I am not validating this a user could just changing the number and access different user accounts (Panera Bread Hack)

How to Prevent

- https://site.com/users/1234
- Need to validate input