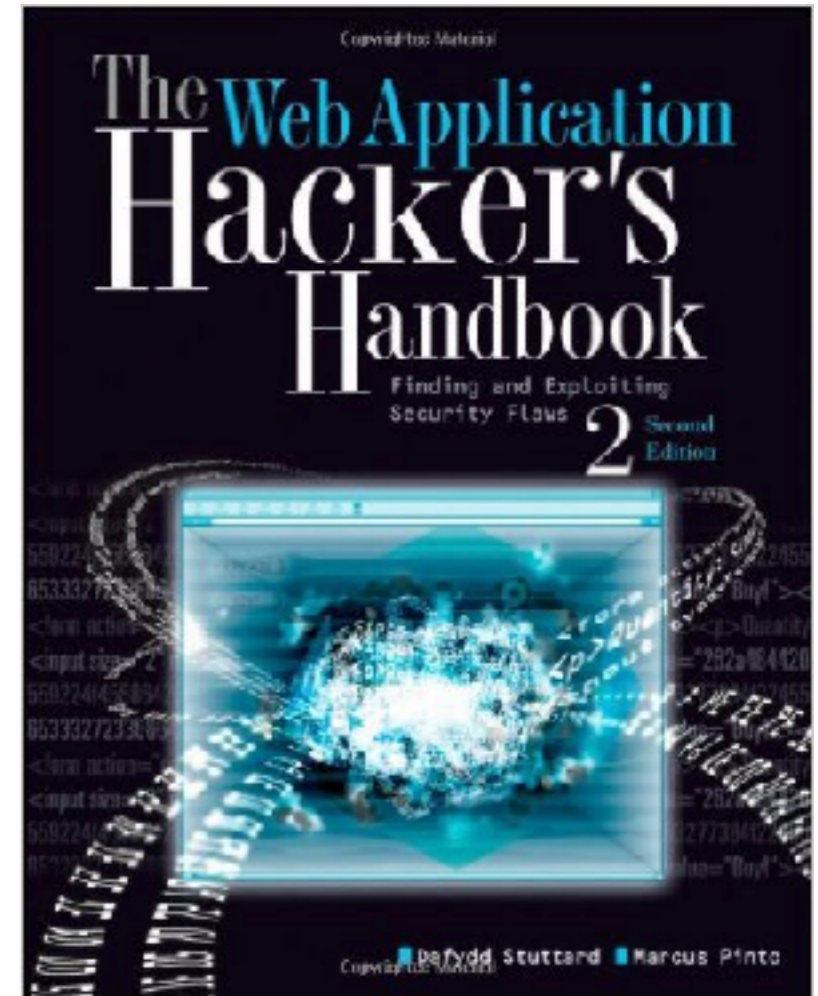


# CNIT 129S: Securing Web Applications

## Ch 1: Web Application (In)security



Updated 1-17-18

# Web Applications

- **E-commerce, Social networks, Online banking, etc.**
- **Fundamental security problem:**
  - *Users can supply arbitrary input*
- **Malicious input can compromise the site**

# Static Website: Web 1.0

- **Information flows one-way**
- **Users don't log in, shop, or submit comments**
- **An attacker who exploits flaws in the Web server software can**
  - **Steal data on the Web server (usually only public data anyway)**
  - **Deface the site**



# Modern Web App

- **Two-way information flow**
- **Users log in, submit content**



- **Content dynamically generated and tailored for each user**
- **Much data is sensitive and private (e.g. passwords)**
- **Most apps developed in-house**
- **Developers often naive about security**

# Common Web App Functions

- **Shopping (Amazon)**
- **Social networking (Facebook)**
- **Banking (Citibank)**
- **Web search (Google)**
- **Auctions (eBay)**
- **Gambling (Betfair)**
- **Web logs (Blogger)**
- **Web mail (Gmail)**
- **Interactive information (Wikipedia)**

# Internal Web Apps ("Cloud" Services)

- **HR -- payroll information, performance reviews**
- **Admin interfaces to servers, VMs, workstations**
- **Collaboration software (SharePoint)**
- **Enterprise Resource Planning (ERP)**
- **Email web interfaces (Outlook Web Access)**
- **Office apps (Google Apps, MS Office Live)**

# Benefits of Web Apps

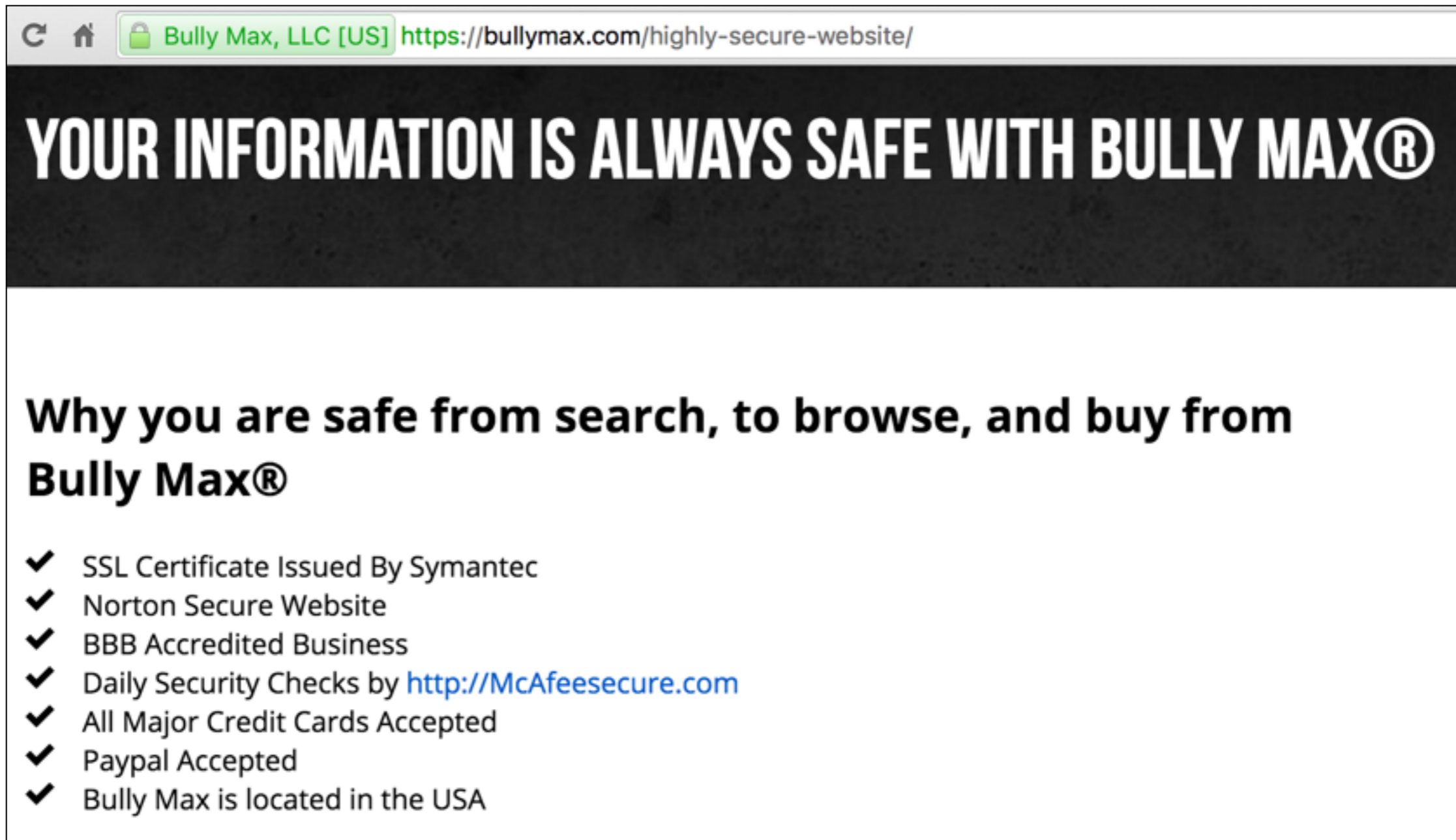
- **HTTP is lightweight and connectionless**
  - **Resilient in event of communications errors**
  - **Can be proxied and tunneled over other protocols**
- **Web browsers run on many devices, highly functional, easy to use**
- **Many platforms and development tools available**

# Web App Security

- **Breaches are common**
- **Attackers gets sensitive data, possibly complete control of back-end systems**
- **Denial of Service at Application Level**



# This Site is Secure



The image is a screenshot of a web browser window. The address bar at the top shows a green padlock icon, followed by the text "Bully Max, LLC [US]" and the URL "https://bullymax.com/highly-secure-website/". Below the address bar is a dark grey banner with the text "YOUR INFORMATION IS ALWAYS SAFE WITH BULLY MAX®" in white, bold, uppercase letters. Underneath the banner, the text "Why you are safe from search, to browse, and buy from Bully Max®" is displayed in bold black font. Below this is a list of seven security features, each preceded by a checkmark icon.

**YOUR INFORMATION IS ALWAYS SAFE WITH BULLY MAX®**

**Why you are safe from search, to browse, and buy from Bully Max®**

- ✓ SSL Certificate Issued By Symantec
- ✓ Norton Secure Website
- ✓ BBB Accredited Business
- ✓ Daily Security Checks by <http://McAfeesecure.com>
- ✓ All Major Credit Cards Accepted
- ✓ Paypal Accepted
- ✓ Bully Max is located in the USA

# 100% Secure Online Voting

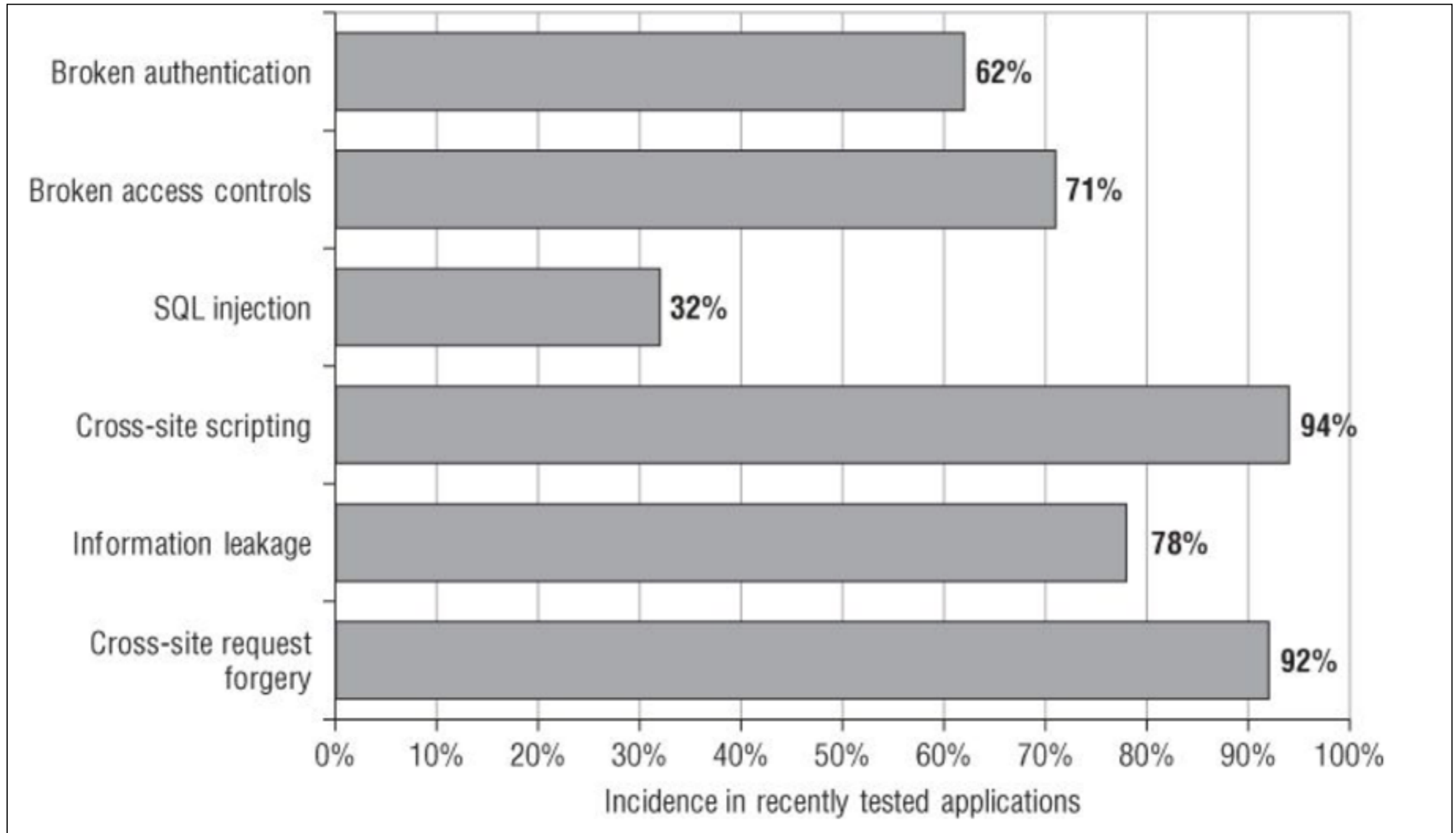
- **Link Ch 1b**



**100% Secure**

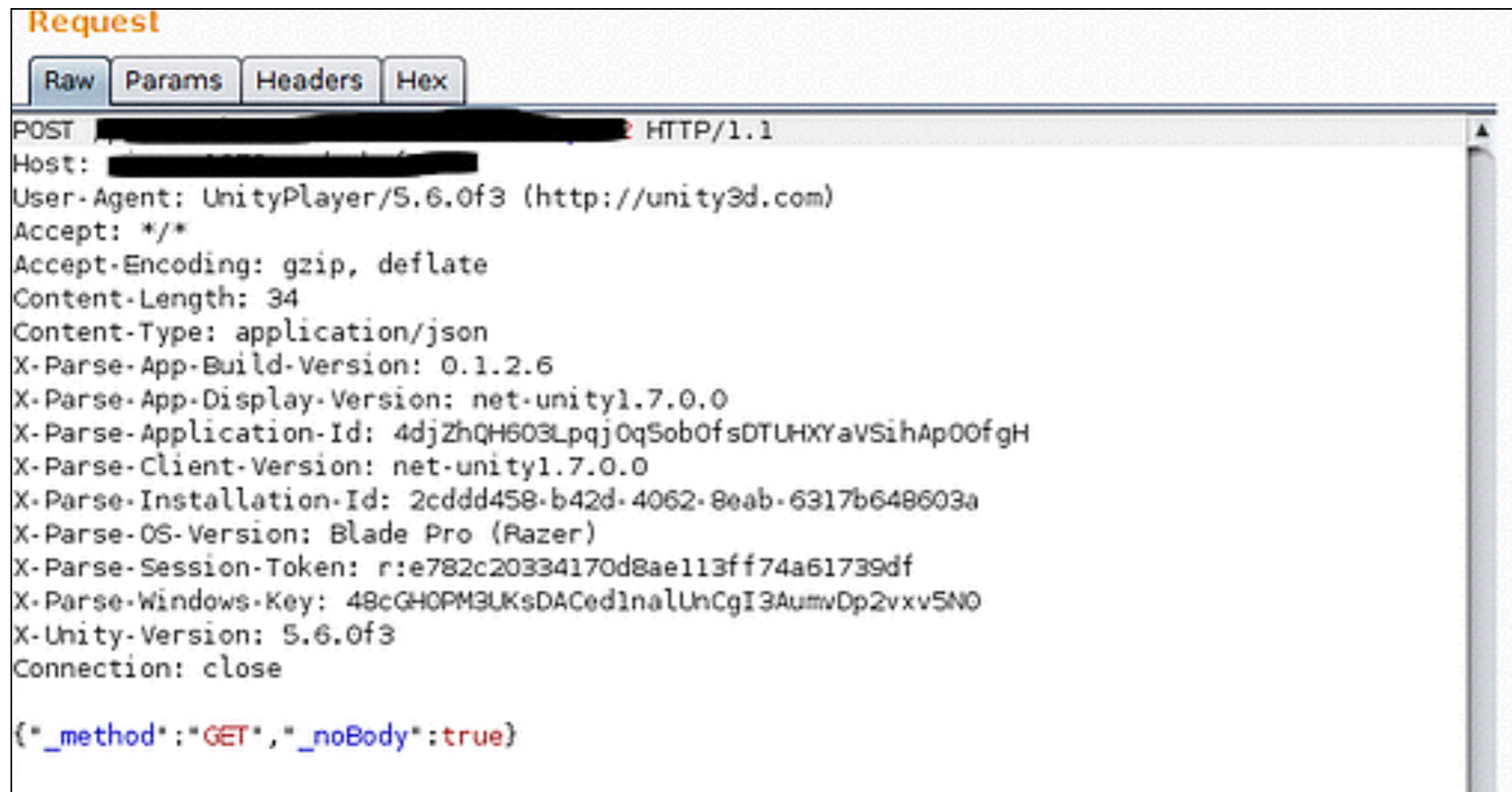
**Blockchain technology** ensures  
that the ballot box cannot be  
hacked.

# Study by Text Authors



# SinVR Hack

- Unauthenticated request
  - Link Ch 1f



The screenshot shows a web browser's developer tools interface with the 'Request' tab selected. The request is a POST method to a redacted URL. The headers include various Unity-specific information such as version, application ID, and session token. The body of the request is a JSON object: {"\_method": "GET", "\_noBody": true}.

```
Request
Raw Params Headers Hex
POST [redacted] HTTP/1.1
Host: [redacted]
User-Agent: UnityPlayer/5.6.0f3 (http://unity3d.com)
Accept: */*
Accept-Encoding: gzip, deflate
Content-Length: 34
Content-Type: application/json
X-Parse-App-Build-Version: 0.1.2.6
X-Parse-App-Display-Version: net-unity1.7.0.0
X-Parse-Application-Id: 4djZhQH603LpqjOqSobOfsDTUHXyaVSihAp00fgH
X-Parse-Client-Version: net-unity1.7.0.0
X-Parse-Installation-Id: 2cddd458-b42d-4062-8eab-6317b648603a
X-Parse-OS-Version: Blade Pro (Razer)
X-Parse-Session-Token: r:e782c20334170d8ae113ff74a61739df
X-Parse-Windows-Key: 48cGHOPM3UKsDACed1naUnCgI3AumvDp2vxv5N0
X-Unity-Version: 5.6.0f3
Connection: close

{"_method": "GET", "_noBody": true}
```

## Response

Raw Headers Hex

```
HTTP/1.1 200 OK
X-Powered-By: Express
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET,PUT,POST,DELETE,OPTIONS
Access-Control-Allow-Headers: X-Parse-Master-Key, X-Parse-REST-API-Key, X-Parse-Javascript-Key, X-Parse-Application-Id,
X-Parse-Client-Version, X-Parse-Session-Token, X-Requested-With, X-Parse-Revocable-Session, Content-Type
Content-Type: application/json; charset=utf-8
ETag: W/"29645-9A769/sy957+0liF6X6eiw"
Vary: Accept-Encoding
Date: Tue, 09 Jan 2018 22:32:46 GMT
Connection: keep-alive
Set-Cookie: nodechefroute=2430159827
Content-Length: 169541
```

```
{*results*:[{*residence_country*:"ES",*first_name*:"██████████",*mc_fee*:"0.69",*notify_version*:"3.8",*payment_fee*:"0.69",*item_name*:"7 Day
Trial, renews to 1 Month SinVR Membership, cancel
anytime",*payer_email*:"██████████@gmail.com",*payment_status*:"Completed",*payment_gross*:"9.99",*updatedAt*:"2016-11-22T22:45:06.763Z",*o
bjectId*:"██████████",*charset*:"windows-1252",*receiver_id*:"██████████",*business*:"eyal@invr.co",*mc_currency*:"USD",*custom*:"██████████
██████████",*txn_id*:"██████████",*ipn_track_id*:"██████████",*protection_eligibility*:"Ineligible",*createdAt*:"2016-11-22T22:45:06.
763Z",*payment_date*:"14:45:01 Nov 22, 2016 PST",*last_name*:"██████████
██████████",*payment_type*:"instant",*payer_status*:"verified",*userPointer*:{*_type*:"Pointer",*className*:"_User",*objectId*:"██████████"},*r
eceiver_email*:"eyal@invr.co",*mc_gross*:"9.99",*payer_id*:"██████████",*transactionData*:"transaction_subject=7+Day+Trial%2C+renews+to
+1+Month+SinVR+Membership%2C+cancel+anytime&payment_date=14%3A45%3A01+Nov+22%2C+2016+PST&txn_type=subscr_payment&subscr_id=██████████&
last_name=██████████&residence_country=ES&item_name=7+Day+Trial%2C+renews+to+1+Month+SinVR+Membership%2C+cancel+anytime&payment_gross=9.99
&mc_currency=USD&business=eyal%40invr.co&payment_type=instant&protection_eligibility=Ineligible&verify_sign=██████████
██████████&payer_status=verified&payer_email=a██████████.40gmail.com&txn_id=██████████&receiver_email=eyal%40invr.co&
first_name=██████████&payer_id=XG9LKSCGAQCE4&receiver_id=██████████&item_number=10&payment_status=Completed&payment_fee=0.69&mc_fee=0.69&
mc_gross=9.99&custom=27AUDYovUs&charset=windows-1252&notify_version=3.8&ipn_track_id=██████████",*verify_sign*:"██████████
██████████",*txn_type*:"subscr_payment",*userObjectId*:"██████████",*transaction_subject*:"7 Day Trial, renews to
1 Month SinVR Membership, cancel
anytime",*item_number*:"10",*subscr_id*:"██████████"}] (*residence_country*:"ES",*first_name*:"██████████",*subscr_id*:"██████████")
```

## OWASP Mobile Top 10 Risks

M1 – Weak Server Side Controls

M2 – Insecure Data Storage

M3 - Insufficient Transport Layer Protection

M4 - Unintended Data Leakage

M5 - Poor Authorization and Authentication

M6 - Broken Cryptography

M7 - Client Side Injection

M8 - Security Decisions Via Untrusted Inputs

M9 - Improper Session Handling

M10 - Lack of Binary Protections

# Android Apps Vulnerable to Code Modification

## Banks



Bank of America

**Bank of America**  
[\(10 Million\)](#)

Notified 2-7-15  
No reply  
Still vulnerable on 5-22-15  
Still vulnerable on 6-14-15

[Details & PoC](#)



The Bancorp

**Bancorp**  
[\(10,000\)](#)

Notified 2-26-15  
No reply  
Last update 4-26-14  
Still vulnerable 5-22-15

[Details & PoC](#)



Capital One

**Capital One**  
[\(5 Million\)](#)

Notified 2-26-15  
No reply  
Still vulnerable on 5-22-15

[Details & PoC](#)



CHASE

**Chase Manhattan**  
[\(10 Million\)](#)

Notified 2-9-15  
Twitter acknowledgement  
Still vulnerable on 5-22-15  
Fixed in 6-8-15 update!

[Details & PoC](#)

- **Link Ch 1c**



<https://samsclass.info/125/proj11/college-security.htm>

# Security Problems at Colleges

I have several of these projects underway. This page is the directory to them so I can keep track of them.

## **A: Viagra Sellers**

A web page redirector that cleverly hides on servers.

[As of 12-4-13, 5/19 colleges are clean](#)

[How I cleaned an infected site on 4-22-16](#)  
[70 Infected Sites \(May, 2016\)](#)

## **B: Exposed Student (or Staff) Data**

As of 12-9-13, 8/12 colleges have fixed this, 6 days after notification.

The vendor (Jenzabar) has patched the problem, and only 3 colleges remain vulnerable, on 7-19-14. I p

## **C: 55 SQLi Vulns Notified in November**

As of 12-4-13, 19/53 colleges have fixed this, 23 days after notification.

- **Link Ch 1d**



# The Core Security Problem

- **Users can submit arbitrary input**
  - **Alter parameters, cookies, HTTP headers**
  - **Client-side controls can't be trusted**
  - **Developers must assume all input is malicious**
- **Attackers have attack tools like Burp; they are not restricted to using browsers**

# Possible Attacks

- **Change the price of an item**
- **Modify a session token to enter another user's account**
- **Remove parameters to exploit logic flaws**
- **SQL injection**
- ***SSL doesn't stop any of these***

# Key Problem Factors

- **Underdeveloped Security Awareness**
- **Custom Development**
- **Deceptive Simplicity**
  - **Easy to make a website, but hard to secure it**
- **Rapidly Evolving Threat Profile**
- **Resource and Time Constraints**
- **Overextended Technologies**
- **Increasing Demands on Functionality**

# The New Security Perimeter

- **Edge firewalls and "bastion hosts" are no longer enough**
  - **Keeping the attacker out of critical systems**
- **Customers can now send transactions to servers holding private data**
  - **Via the Web app**
- **Web app must act as a security barrier**
- **Often it includes components from others, like widgets**
- **Errors by other companies can compromise your servers**

# The New Security Perimeter

- **Attackers can attack users instead of servers**
  - **XSS, drive-by downloads, etc.**
- **E-mail used for password recovery**
  - **A compromised email account exposes many other services also**

# The Future

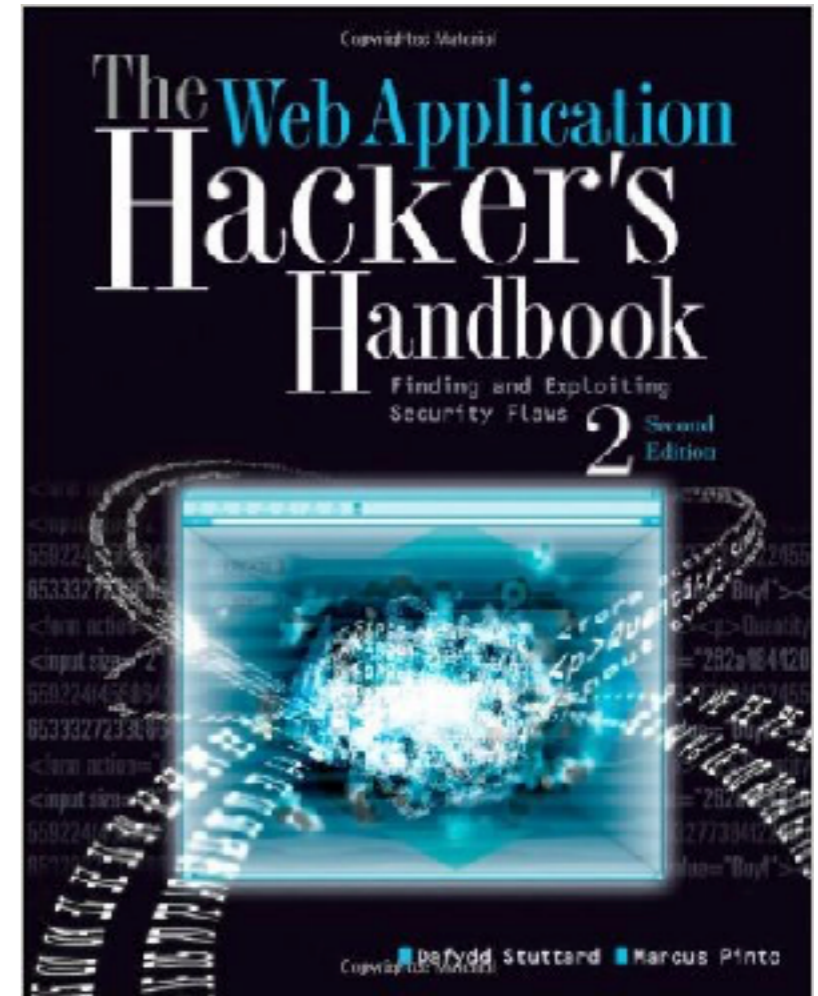
- **Some vulnerabilities are decreasing**
  - **#1 security measure: UPDATES**
- **Logic flaws and failure to use controls properly are not decreasing**
- **Link Ch 1e**

Year	# of Vulnerabilities	DoS	Code Execution	Overflow	Memory Corruption	Sql Injection
<a href="#">2004</a>	2					
<a href="#">2005</a>	10		<a href="#">5</a>			<a href="#">3</a>
<a href="#">2006</a>	16	<a href="#">1</a>	<a href="#">2</a>			<a href="#">1</a>
<a href="#">2007</a>	48	<a href="#">2</a>	<a href="#">13</a>			<a href="#">7</a>
<a href="#">2008</a>	55	<a href="#">2</a>	<a href="#">20</a>			<a href="#">15</a>
<a href="#">2009</a>	16	<a href="#">3</a>	<a href="#">1</a>			
<a href="#">2010</a>	2		<a href="#">1</a>			<a href="#">1</a>
<a href="#">2011</a>	12		<a href="#">1</a>			<a href="#">2</a>
<a href="#">2012</a>	32	<a href="#">2</a>	<a href="#">4</a>			<a href="#">3</a>
<a href="#">2013</a>	20	<a href="#">1</a>	<a href="#">1</a>			
<a href="#">2014</a>	29	<a href="#">3</a>	<a href="#">3</a>			<a href="#">1</a>
<a href="#">2015</a>	11	<a href="#">1</a>	<a href="#">2</a>			<a href="#">1</a>
<a href="#">2016</a>	17	<a href="#">1</a>				
Total	270	<a href="#">16</a>	<a href="#">53</a>			<a href="#">34</a>

**Kahoot!**

# CNIT 129S: Securing Web Applications

## Ch 2: Core Defense Mechanisms





# Core Defense Elements

- **Limiting *user access* to app's data and functionality**
- **Limiting *user input* to prevent exploits that use malformed input**
- ***Frustrating attackers* with appropriate behavior when targeted**
- ***Administrative monitoring and configuring* the application**

# Handling User Access


- **Authentication**
- **Session management**
- **Access Control**

# Authentication

- **Username and password is most common method**
- **Better: additional credentials and multistage login**
- **Best: client certificates, smart cards, challenge-response tokens**
- **Also: self-registration, account recovery, password change**

**Figure 2.1** A typical login function

## Log in



Please log in below by completing the details requested, then select 'Log In'.

For security reasons, you have a limited number of attempts to provide the correct information. If you do not provide the correct information, access to your Intelligent Finance plan will be suspended. If this happens, please call **0845 609 4343** and we will send you a new Plan Security Code. You will then be able to access your plan by following the [reactivation process](#).

If you are not sure about your login details or require help, please call us.

<b>Online Username</b>	<input type="text"/>	<i>This must be at least 6 characters long and can have letters and / or numbers, but no spaces.</i>
<b>Online Password</b>	<input type="password"/>	<i>This must be at least 6 characters long and must have both letters and numbers, but no spaces.</i>

**Log In**

# Common Login Problems

- **Predictable usernames**
- **Password that can be guessed**
- **Defects in logic**

# Session Management

- **Session: a set of data structures that track the state of the user**
- **A token identifies the session, usually a cookie**
  - **Can also use hidden form fields or the URL query string**
- **Sessions expire**

**Your Account Session has ended**\_\_\_\_\_

Sorry - for your own protection we have had to log you out of your online account because you did not use the service for more than 10 minutes. To re-enter your account, please log in again.

Would you like to log in now?

# Common Session Problems

- **Tokens are predictable (not random)**
- **Tokens poorly handled, so an attacker can capture another user's token**

# Access Control

- **Each request must be permitted or denied**
- **Multiple roles within application**
- **Frequent logic errors and flawed assumptions**

[Home](#) » [Access Denied \[403\]](#)

## Access Denied [403]

---

We're sorry...

You are not authorized to access this page.

- [Login](#) to the site.
- If you typed the page url, check the spelling.
- Click your browser's back button and try another link.
- Consider [telling us](#) about the broken link that led you to this page.

We apologize for the inconvenience, and hope we'll see you again soon.



# Handling User Input

- **"Input Validation" is the most common solution**

First Name	<input type="text" value="a"/>	Must contain at least 4 characters
Last Name	<input type="text" value="a"/>	Must contain at least 4 characters
Email	<input type="text" value="a"/>	Please provide a valid email address
Phone number	<input type="text" value="a"/>	Must contain only numbers

# Types of Input

- **Arbitrary text, like blog posts**
- **Cookies**
- **Hidden form fields**
- **Parameters**
- **HTTP header fields, like User-Agent**

# "Reject Known Bad"

- **Also called "blacklisting"**
- **Least effective method**
- **Difficult to identify all bad inputs**

- If `SELECT` is blocked, try `seLeCt`
- If `or 1=1--` is blocked, try `or 2=2--`
- If `alert('xss')` is blocked, try `prompt('xss')`

```
SELECT/*foo*/username,password/*foo*/FROM/*foo*/users  
<img%09onerror=alert(1) src=a>
```

```
%00<script>alert(1)</script>
```

# "Accept Known Good"

- **Also called "whitelisting"**
- **Most effective technique, where feasible**
- **However, sometimes you can't do it**
  - **Human names really contain apostrophes, so you can't filter them out**

# Sanitization

- **Render dangerous input harmless**
- **HTML-Encoding: Space becomes %20, etc.**
- **Difficult if several kinds of data may be present within an item of input**
- **Boundary validation is better (four slides ahead)**

# Safe Data Handling

- **Write code that can't be fooled by malicious data**
  - **SQL parameterized queries**
  - **Don't pass user input to an OS command line**
- **Effective when it can be applied**

# Semantic Checks

- **Some malicious input is identical to valid input**
  - **Such as changing an account number to another customer's number**
- **Data must be validated in context**
  - **Does this account number belong to the currently logged-in user?**

# Difficulties with Simple Input Validation

- **Data coming from user is "bad" or "untrusted"**
- **The server-side app is "good" and trusted**
  - **Many different types of input with different filtering requirements**
  - **Apps may chain several processing steps together**
    - **Data may be harmless at one stage, but be transformed into harmful data at another stage**



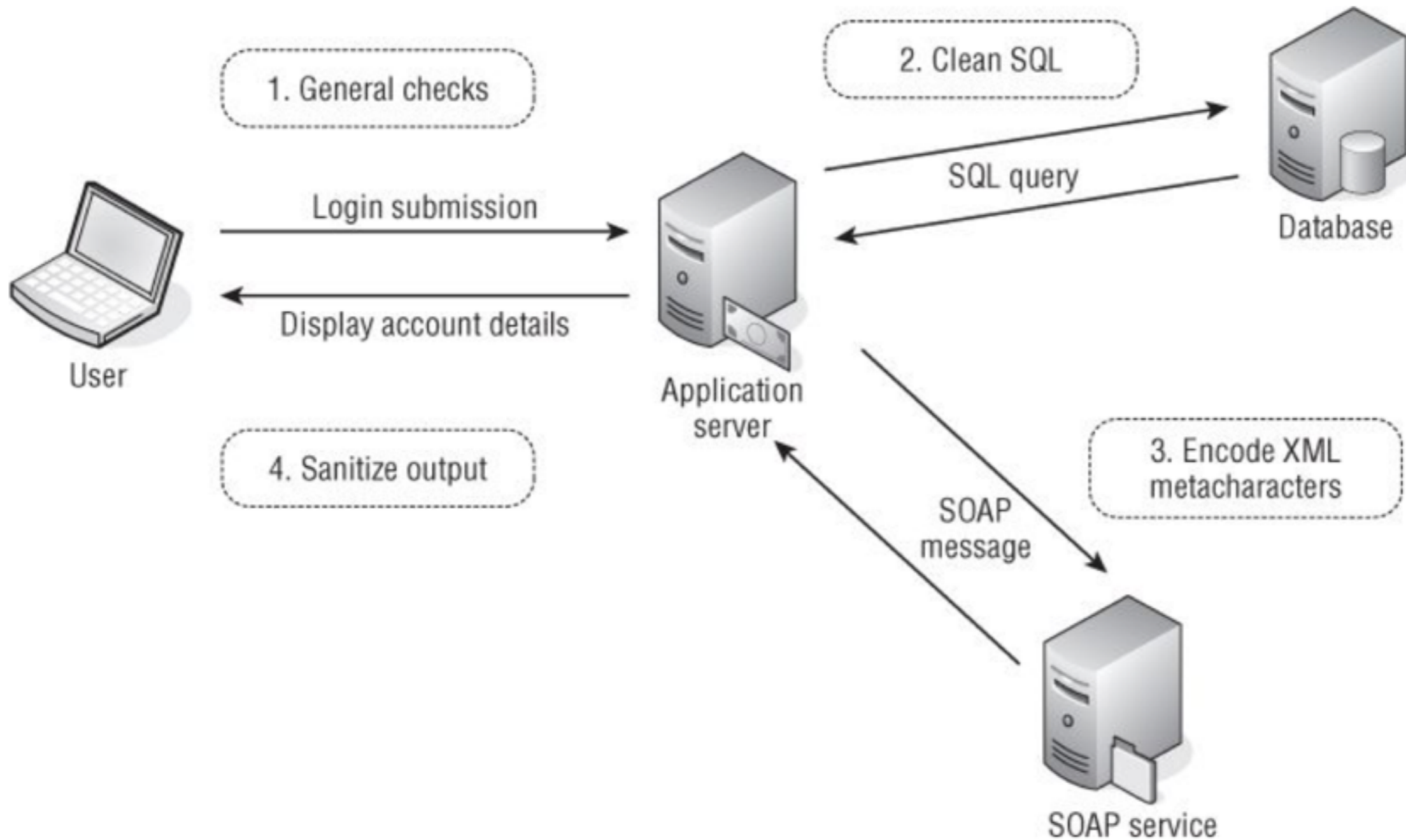
# Boundary Validation

- **Trust boundary**
  - **Divides a trusted zone from an untrusted zone**
- **Clean data that passes a boundary**
  - **Such as from the user into an application**

# Boundary Validation

- **Each component treats its input as potentially malicious**
- **Data validation performed at each trust boundary**
  - **Not just between client and server**

# Example



# Example SOAP Request

```
POST /Quotation HTTP/1.0
Host: www.xyz.org
Content-Type: text/xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2001/12/soap-envelope"
SOAP-ENV:encodingStyle="http://www.w3.org/2001/12/soap-encoding" >

  <SOAP-ENV:Body xmlns:m="http://www.xyz.org/quotations" >

    <m:GetQuotation>
      <m:QuotationsName>MiscroSoft</m:QuotationsName>
    </m:GetQuotation>

  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

- **Link Ch 2a**

# Boundary Validation Example

- **1. App gets login: username and password**
  - **Allows only good characters, limits length, removes known attack signatures**
- **2. App performs a SQL query to verify credentials**
  - **Escape dangerous characters**

# Boundary Validation

## Example

- **3. Login succeeds; app passes data from user profile to a SOAP service**
  - **XML metacharacters are encoded to block SOAP injection**
- **4. App displays user's account information back to the user's browser**
  - **User-supplied data is HTML-encoded to block XSS**

# Filtering Problems

- **App removes this string:**
  - **<script>**
- **So attacker sends this**
  - **<scr<script>ipt>**

# Multistep Validation

- **App first removes**

.. /

- **Then removes**

.. \

- **Attacker sends**

... \



# Canonicalization

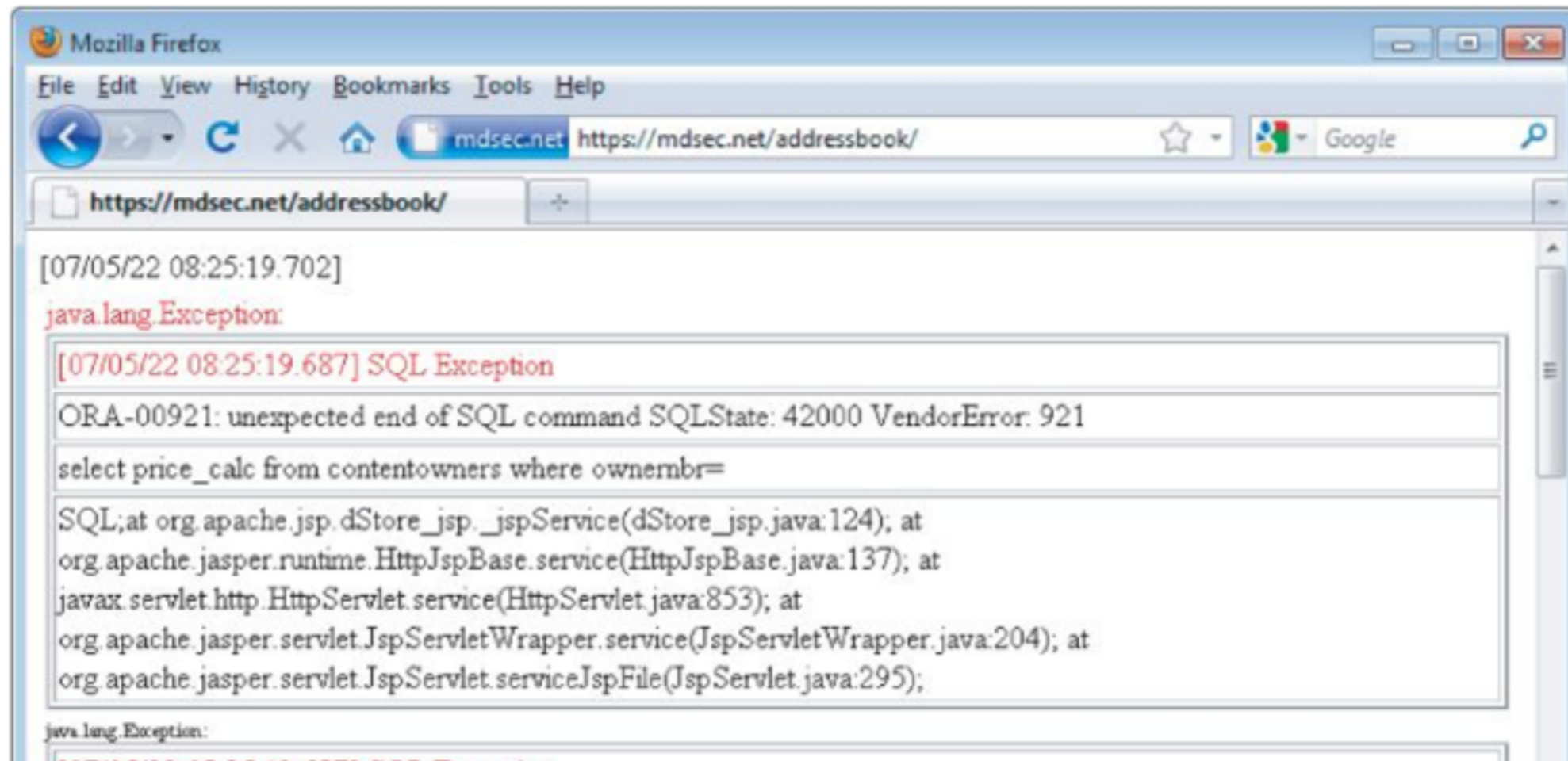
- **App gets URL-encoded data from Web browser**
  - **Apostrophe is %27**
  - **Percent is %25**
- **To block apostrophes, app filters %27**
- **But URL is decoded twice by mistake**
  - **%2527 becomes %27 becomes apostrophe**

# Handling Attackers

- **Handling errors**
- **Maintaining audit logs**
- **Alerting administrators**
- **Reacting to attacks**

# Handling Errors

- **Show appropriate error messages**
- **Unhanded errors lead to overly-informative error messages like this**



# Audit Logs

- **Authentication events: login success and failure, change of password**
- **Key transactions, such as credit card payments**
- **Access attempts that are blocked by access control mechanisms**
- **Requests containing known attack strings**
- **For high security, log every client request in full**

# Protecting Logs

- **Logs should contain time, IP addresses, and username**
  - **May contain cookies and other sensitive data**
- **Attackers will try to erase and/or read logs**
- **Log must be protected**
  - **Place on an autonomous system that only accepts update messages**
  - **Flush logs to write-once media**

# Alerting Administrators

- **Usage anomalies, like a large number of requests from the same IP address or user**
- **Business anomalies, such as a large number of funds transfers to/from the same account**
- **Requests containing known attack strings**
- **Requests where hidden data has been modified**

# Firewalls

- **Web App Firewalls can detect generic attacks**
  - **But not subtle ones that are specific to your app**
- **Most effective security control is integrated with app's input validation mechanisms**
  - **Check for valid values of your parameters**

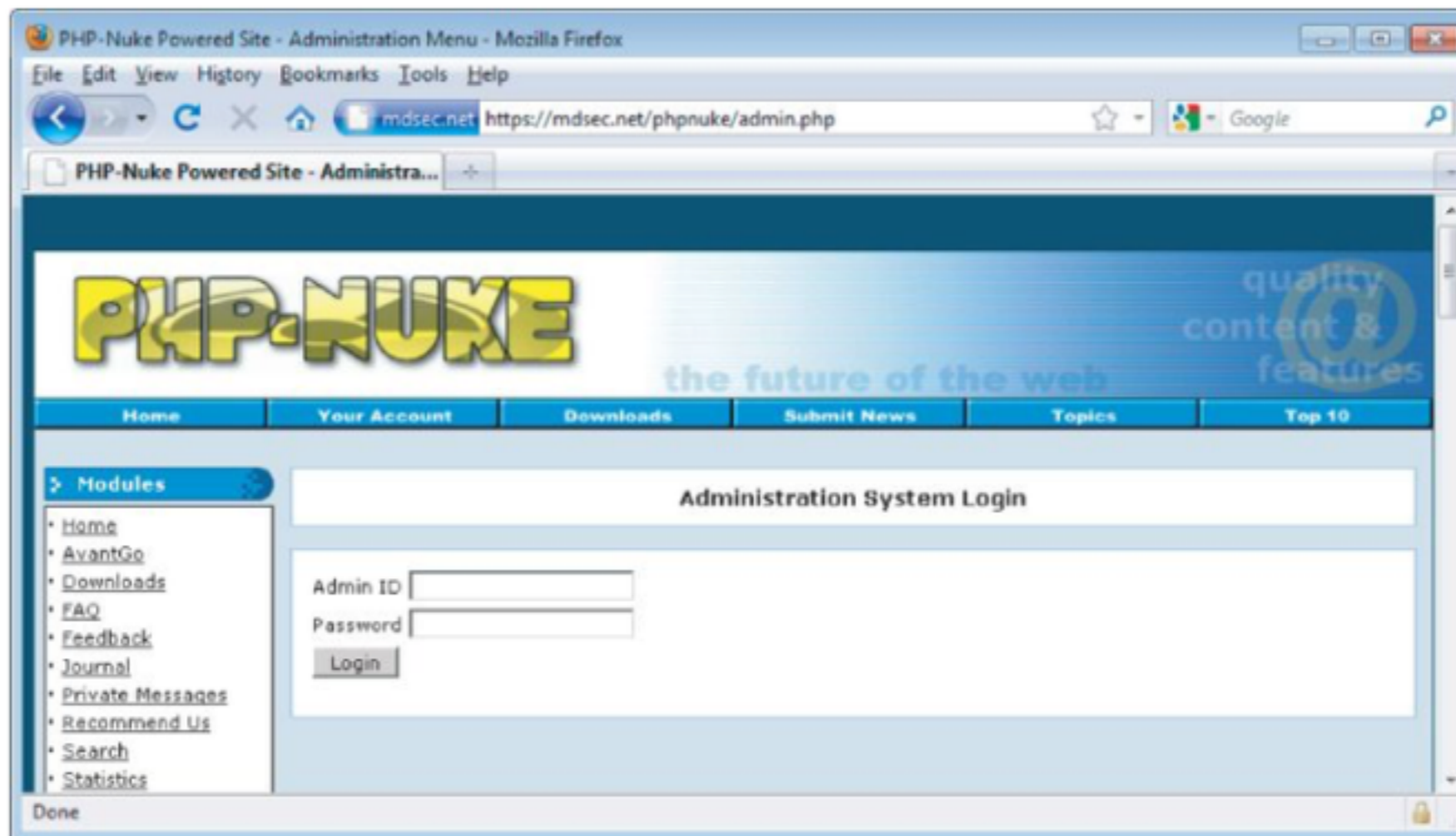
# Reacting to Attacks

- **Attackers probe for vulnerabilities**
- **Sending many similar requests**
- **Automated defenses**
  - **Respond increasingly slowly to requests**
  - **Terminate the attacker's session**



# Managing the Application

- **Management interface allows administrator to control user accounts, roles, access monitoring, auditing, etc.**



# Attacking the Administration Panel

- **Defeat weak authentication**
- **Some administrative functions might not require high privileges**
- **XSS flaws can allow cookie theft and session hijacking**

**Kahoot!**