

Objectives

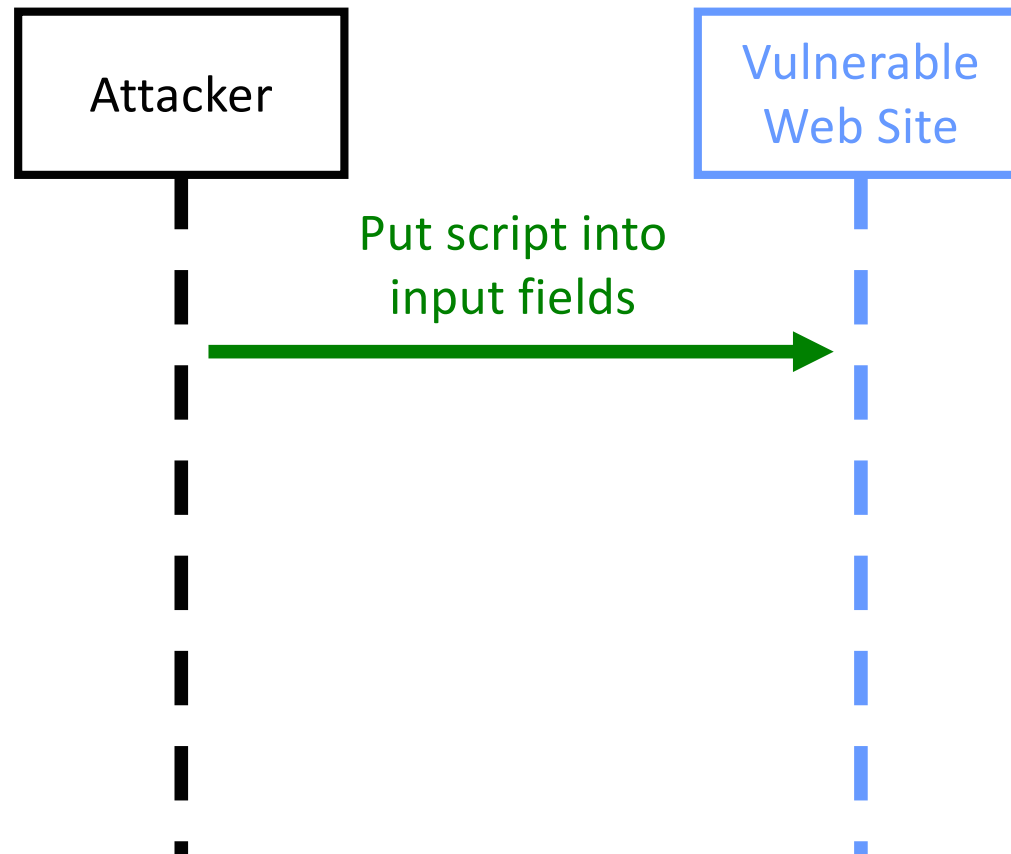
- Security
 - Cross-site scripting
 - Evaluating
- Project

Review: Security

- Why is the Web such a huge target?
- What is https?
- What are some security design principles?
 - Provide examples of their use/application
- What is an SQL Injection Attack?
 - How can you prevent against it?

CROSS-SITE SCRIPTING

Sequence Diagram of a Typical XSS Attack (1)



Unvalidated Input with XSS (1)

Investments Feedback Customer Care Contact

Online Application

Personal Information

An asterisk (*) indicates a required field.

* First Name
(Do not use nicknames)

Middle Initial

* Last Name

* Social Security Number
(format: xxx-xx-xxxx)

* Birth Date
(format yyyy-mm-dd)

* Mother's Maiden Name
(For security verification)

* Address
Unvalidated Input (XSS)

Apartment/Room Number

* City

* State

* Zip Code

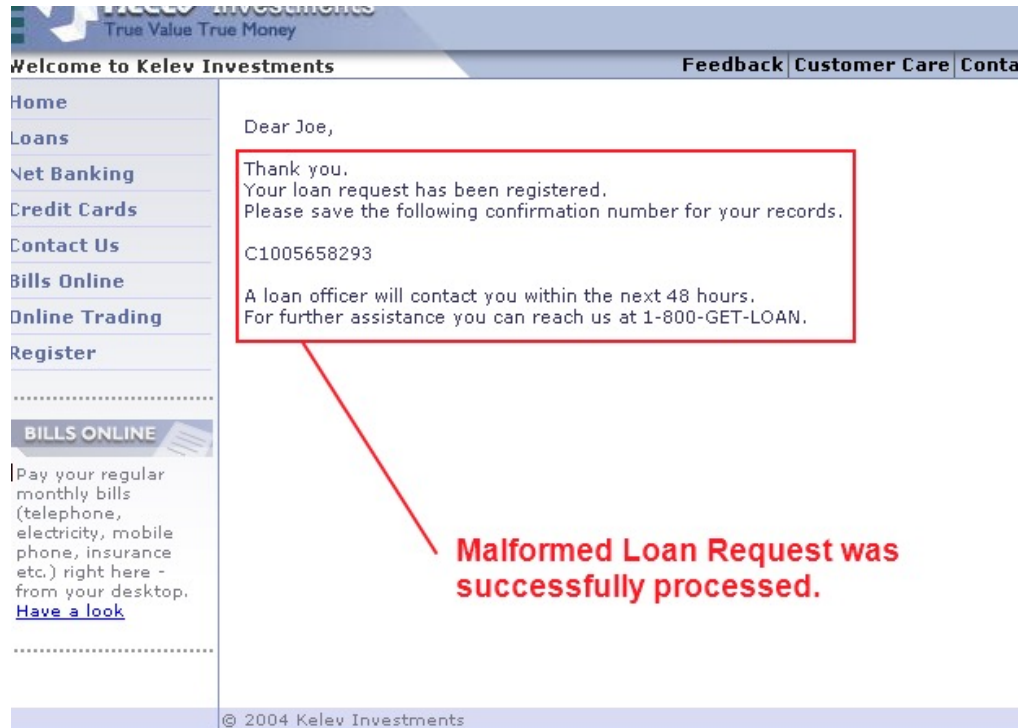
Telephone Number

* Email

Occupation

Annual Income

Unvalidated Input with XSS (1)



True Value True Money
Welcome to Kelev Investments [Feedback](#) [Customer Care](#) [Conta](#)

Home
Loans
Net Banking
Credit Cards
Contact Us
Bills Online
Online Trading
Register

BILLS ONLINE

Pay your regular monthly bills (telephone, electricity, mobile phone, insurance etc.) right here - from your desktop. [Have a look](#)

Dear Joe,

Thank you.
Your loan request has been registered.
Please save the following confirmation number for your records.

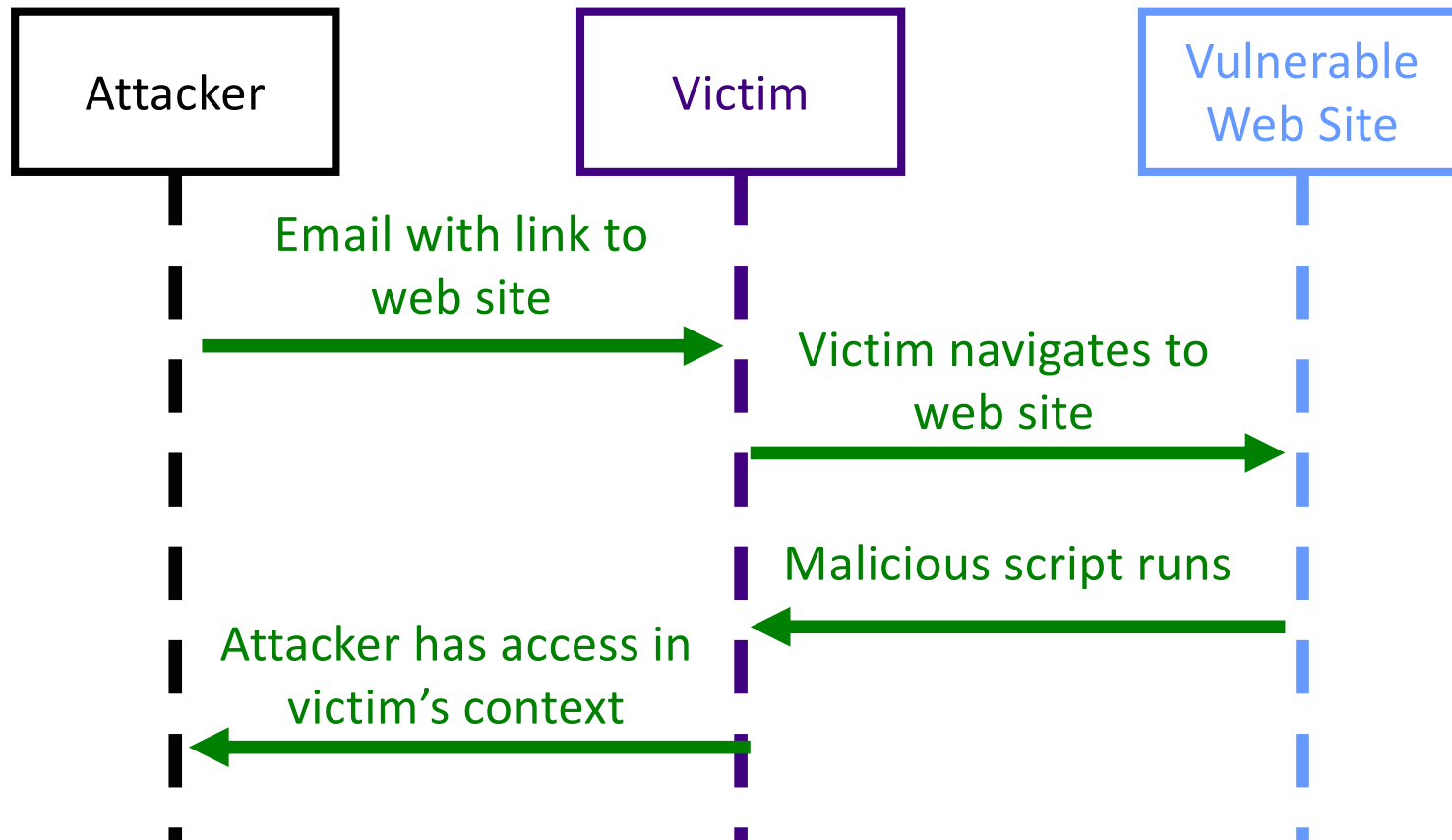
C1005658293

A loan officer will contact you within the next 48 hours.
For further assistance you can reach us at 1-800-GET-LOAN.

Malformed Loan Request was successfully processed.

© 2004 Kelev Investments

Sequence Diagram of a Typical XSS Attack (2)

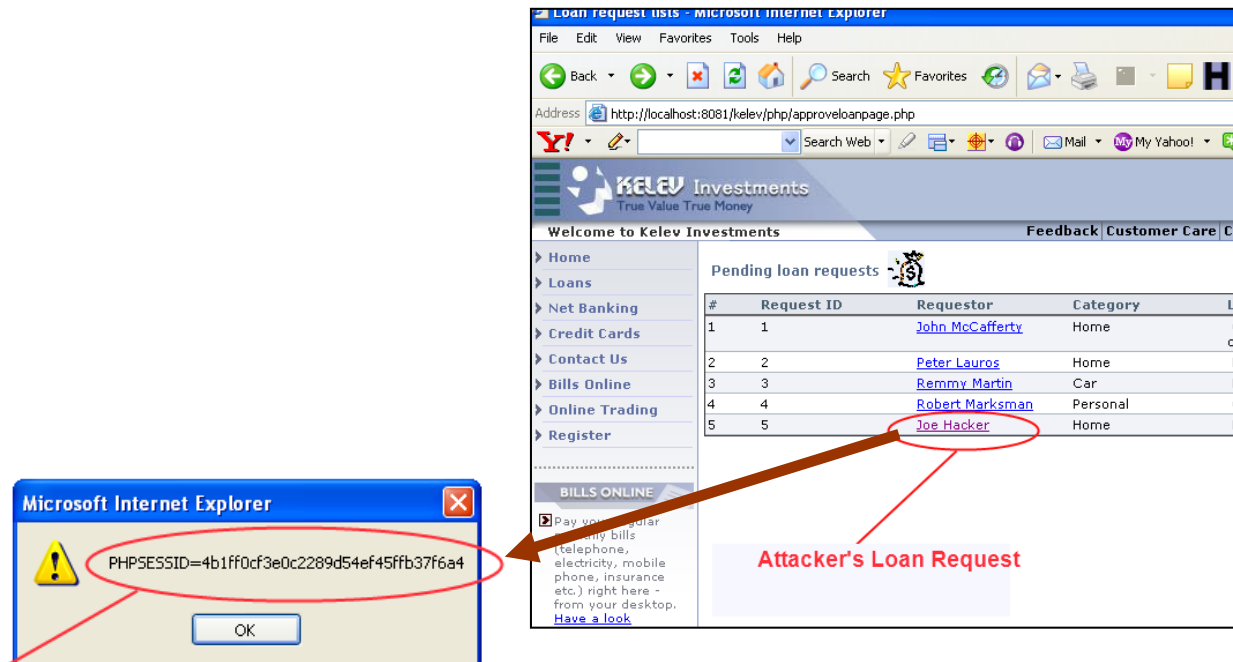


Unvalidated Input with XSS

The screenshot shows a web browser window with the address `http://localhost:8081/kelev/php/approveloanpage.php`. The page title is "Loan Request Lists - Microsoft Internet Explorer". The browser's address bar and toolbar are visible. The page content includes a navigation menu on the left, a header for "KELEV Investments", and a table of pending loan requests. The table has columns for "#", "Request ID", "Requestor", "Category", and "Loan". The "Requestor" column contains the names of the requesters. The name "Joe Hacker" is circled in red, and a red arrow points from this circle to a text box labeled "Attacker's Loan Request".

#	Request ID	Requestor	Category	Loan
1	1	John McCafferty	Home	Of che
2	2	Peter Lauros	Home	Ba
3	3	Remmy Martin	Car	Mi
4	4	Robert Marksman	Personal	Cc
5	5	Joe Hacker	Home	Ha

Unvalidated Input with XSS



Unvalidated Input resulted in a Cross-Site Scripting Attack and theft of administrator's cookie.

- Attacker would probably inject some other script, not actually a popup

Cross-Site Scripting (XSS)

Occurs any time...

- Raw data from attacker is sent to an innocent user's browser

Raw data...

- Stored in database
- Reflected from web input (form field, hidden field, URL, etc...)
- Sent directly into rich JavaScript client

Typical Impact

- Steal user's session, steal sensitive data, rewrite web page, redirect user to phishing or malware site
- Most Severe: Install XSS proxy which allows attacker to observe and direct all user's behavior on vulnerable site and force user to other sites



Cross-Site Scripting (XSS)

- Cross-site scripting is possible when
 - An adversary tricks a victim into clicking a link crafted and presented to the victim via a web server or email
 - The link contains a URL with embedded malicious script (typically as a query string, for example “phishing”)
 - The URL refers to host that echoes input back to a browser without input validation
- When victim clicks link, goes to the host in the URL
 - Host processes the query string, echoes it to victim's browser
 - Victim's browser executes the malicious script
- **Root Cause:** Failure to proactively reject or scrub malicious characters from input vectors

Cross-Site Scripting (XSS)

- Allows cookie theft, credential theft, data confidentiality, integrity, and availability risks
 - Browser Hijacking and Unauthorized Access to Web Application is also possible using existing exploits
- Unusual vulnerability because the system at fault, i.e., the web site not validating input, is *not* the victim of attack
- Remedy for XSS: web site perform **adequate input validation**
 - Global policy, Form- and Field- specific policies for handling untrusted content

Testing for XSS

- Test for valid HTML and script code allowed in an input field
 - Special characters like `<` or `>`
 - `<script>alert("XSS");</script>`
 - `<script>alert(document.cookie);</script>`
 - `article.php?title=<meta%20http-equiv="refresh"%20content="0;">`
 - Causes denial of service
- Reference:
 - https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

Validating Input

- **Block list:** a list of input types that are expressly forbidden from being used as application input
- **Allowed list:** a list of input types that are expressly allowed as application input
- Generally expressed as **regular expressions**
- Input validation must be server side
 - Not (only) in JavaScript

PROJECT

Developer Tools: Mobile Devices

- Simulate mobile devices with “device mode”

Requirements, Design, and Work Plan

- Requirements
- Design: steps to complete project
 - Includes what will be implemented and the technologies used to implement each piece
- Work Plan: a tentative plan for what parts of the work each member is charged with doing
 - Prioritization of features

Requirements Gathering

- Clarification of requirements
- Involves asking lots of questions
- Talk through the application
 - Flow chart of what happens

Requirements Gathering: Questions

- What does the user want to do?
 - Go through a variety of *use cases*
 - Common case, error case
 - Part of your job is organizing these use cases
- What is needed to do that task?
 - User input? Saved data? Other sources?
- What does the user see?
 - Draw on whiteboard, use paper
 - What is interface?

Static Mock Ups

- High-fidelity prototypes
 - Look like final product but aren't functional
 - May “fake” the flow, e.g., link to a static page rather than a dynamically generated page.
- For us, often not completely static
 - Build from the AGP framework

Varying Projects

- Sizes, difficulty, technologies, ...
- Other tasks to supplement the “main” project

Exam

Projects

- Merge development into your (local) branches