Embedding of External Content from Non-trusted Sources

Alexandre Miguel Ferreira

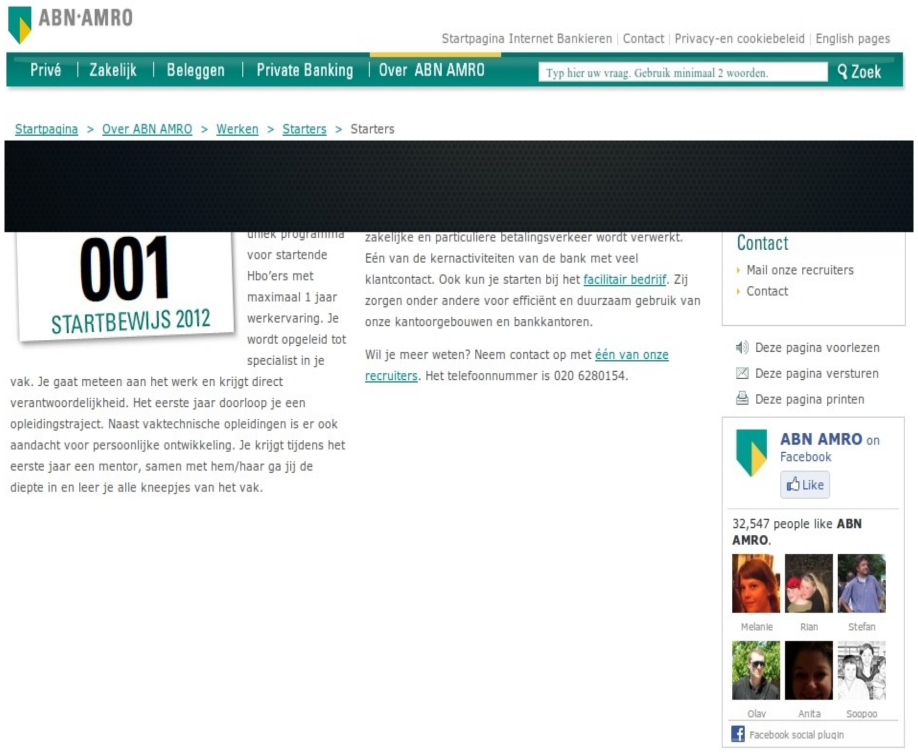
University of Amsterdam
Research Project II

July 5, 2012

Agenda

- Introduction
  - Research Question
- Background
  - How to embed content?
  - Most common attacks
- Results
  - Testing Methods
  - Possible Solutions
- Conclusions
  - Futures work
- Questions

Introduction



ABN·AMRO

Startpagina Internet Bankieren | Contact | Privacy-en cookiebeleid | English pages

Privé | Zakelijk | Beleggen | Private Banking | Over ABN AMRO | Typ hier uw vraag. Gebruik minimaal 2 woorden. | Zoek

Startpagina > Over ABN AMRO > Werken > Starters > Starters

001 STARTBEWIJS 2012

uniek programma voor startende Hbo'ers met maximaal 1 jaar werkervaring. Je wordt opgeleid tot specialist in je vak. Je gaat meteen aan het werk en krijgt direct verantwoordelijkheid. Het eerste jaar doorloop je een opleidingstraject. Naast vaktechnische opleidingen is er ook aandacht voor persoonlijke ontwikkeling. Je krijgt tijdens het eerste jaar een mentor, samen met hem/haar ga jij de diepte in en leer je alle kneepjes van het vak.

zakelijke en particuliere betalingsverkeer wordt verwerkt. Eén van de kernactiviteiten van de bank met veel klantcontact. Ook kun je starten bij het facilitair bedrijf. Zij zorgen onder andere voor efficiënt en duurzaam gebruik van onze kantoorgebouwen en bankkantoren.

Wil je meer weten? Neem contact op met één van onze recruiters. Het telefoonnummer is 020 6280154.

Contact
- Mail onze recruiters
- Contact

Deze pagina voorlezen
Deze pagina versturen
Deze pagina printen

ABN AMRO on Facebook
Like

32,547 people like **ABN AMRO.**

Melanie Rian Stefan
Olav Anita Soopoo

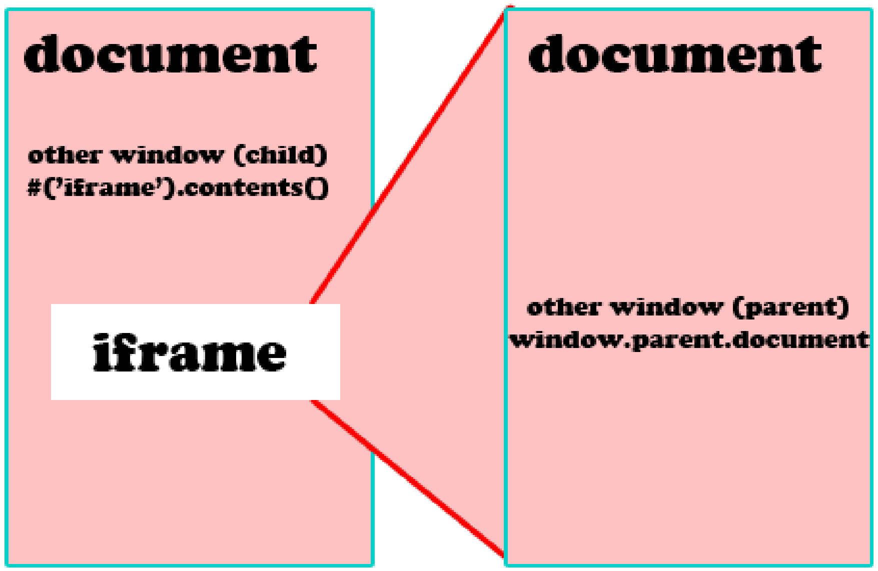
Facebook social plugin

- Target websites
  - e-banking
  - e-commerce
- Embedded third-parties content
  - Bank partners advertising
  - Social networks
- Not all on the same trusted degree!

Introduction
Research Question

- **How to securely embed content from non-trusted sources on a website?**
  - How to create trusted content from untrusted content?
  - Which vulnerabilities have to be secured?
  - How do different browsers handle the problem?
  - How much user intervention is required for the different solutions?
  - What can be secured by the bank server?
  - What can the bank do to secure third parties' servers?
  - What can be done to have a third party to be considered trusted?

Background
How to embed content?



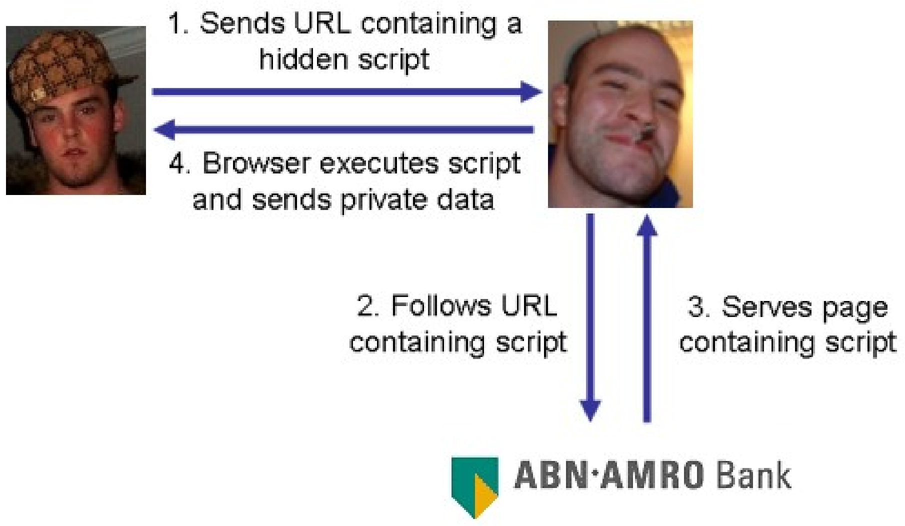
- Content can be included with:
  - *Scripts → <script type="text/javascript">ajaxinclude ("filename.html")</script>*
  - *Inline frames → <iframe src=" https://www.os3.nl/"></iframe>*
- *What is an Iframe?*
  - *HTML document embedded inside another HTML document on a website*
  - *Behaves as an inline image, but can be configured independently from HTML content where it is embed*
  - *More secure than scripts*

Background
Most common attacks{1}

- Cross-site Scripting
  - OWASP Top Ten Project 2010 (A2)
- Cross-site Request Forgery
  - OWASP Top Ten Project 2010 (A5)
- Phishing
  - One of the highest visibility problems for e-banking and e-commerce websites

Background

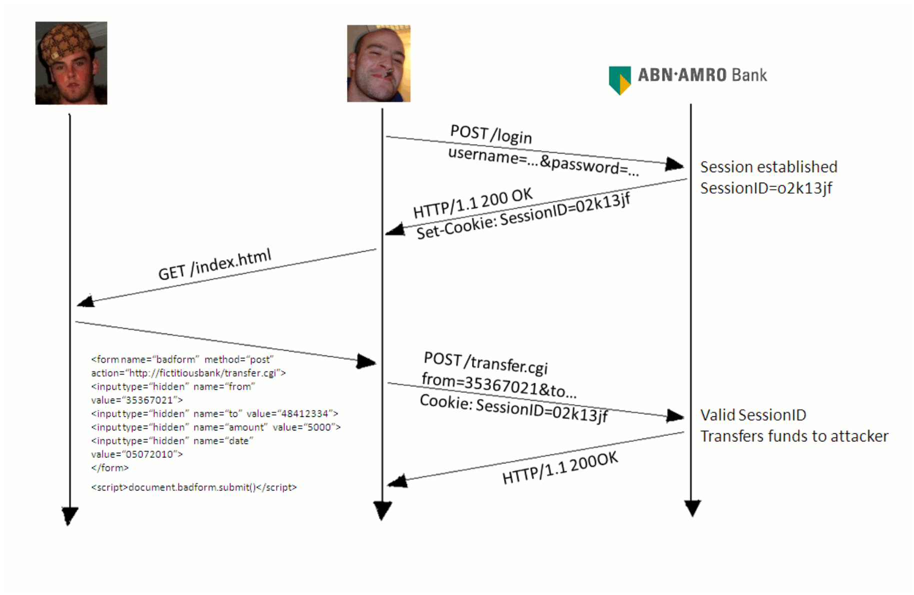
Most common attacks{2}


1. Sends URL containing a hidden script
4. Browser executes script and sends private data
2. Follows URL containing script
3. Serves page containing script
ABN·AMRO Bank

- Cross-site Scripting (XSS)
  - Allow attackers to execute malicious JavaScript code, pretending that the application is sending the code to the user
  - Attacker is able to execute scripts in the victims browser which can be used to hijack users sessions, among others

Background

Most common attacks{3}



```
<form name="badform" method="post"
action="http://fictitiousbank/transfer.cgi">
<input type="hidden" name="from"
value="35367021">
<input type="hidden" name="to" value="48412334">
<input type="hidden" name="amount" value="5000">
<input type="hidden" name="date"
value="05072010">
</form>
<script>document.badform.submit()</script>
```

- Cross-site Request Forgery (CSRF)
  - Allows an attacker to send requests on behalf of a client without knowledge or interaction from the client
  - Attacker can force the victims browser to perform a hostile action, benefiting from this

Background
Most common attacks{4}

- Phishing
  - Good example of social engineering
  - Attacker attempts to obtain informations about the user by misleading him/her
  - Done by masquerading as a trustworthy entity (the bank in this case)

Results
Testing Methods

- Banking website simulated with some flaws
- Inclusion of tree Iframes with attacks to the website
  - **XSS attack** – Session hijacking by stealing cookies
  - **CSRF attack** – Clickable link that will do a POST request, on behalf of the user, to do a new transaction
  - **Phishing attack** – Request to change the user's password
- Three web browsers tested:
  - Firefox
  - Google Chrome
  - Internet Explorer 8

Results

Possible Solutions

- Web Browsers’ Security
- Server-side protections
- Autommated scanners

Results

Possible Solutions – Web Browsers' Security

Web browser/Attack	**XSS**	**CSRF**	**Phishing**
Firefox	Same-origin policy protection	*Use of add-ons such as: CsFire* RequestPolicy* NoScript**	*Phishing Protection* feature*
Google Chrome	Same-origin policy protection	*HTML5 JavaScript Sandbox*	*"Enable phishing and malware protection"* option*
Internet Explorer 8	Same-origin policy protection		*SmartScreen Filter**

* User intervention required

Results
Possible Solutions – Server-side Protection

- XSS not tested (**tested** web browsers handled it)
- CSRF protections
  - Filtering proxy
  - Double submit (variation of the token identification scheme)
  - Apache mod_security module (can be called web application firewall)
- Phishing protections
  - Nothing can be done by server-side!
  - Alert costumers is the best thing to do!

Results
Possible Solutions – Automated Scanners

- Scans the website for malicious content
- It was considered, but …
- … cannot be considered as protection
  - Attacks can be performed in such a way that it can be misled
  - It would only function as a problem detection
- Can be a solution to transform untrusted content into trusted content
  - … but then again it can be misled

Conclusions

- Ideally all the vulnerabilities should be protected (XSS, CSRF and Phishing most common)
- All the **tested** web browsers are protected against XSS (same-origin policy)
- Most of web browsers' features require user intervention
- Phishing is probably the most difficult vulnerability to prevent
- The use of automated scanners can be a solution to transform untrusted content into trusted content, though filtering proxies might do a better job
- CSRF difficult to be protected by web browsers, server side solutions (filtering proxies or double submit) are better
- In order to protect third parties' servers, the same protection methods used by the bank should be used
- Having third parties being audited by the bank should be enough to consider them more trustuble

Conclusions
Future Work

- More web browsers tested
  - Opera
  - Safari
  - Android
- More attacks tested
  - Pharming
  - Man-in-the-Browser (MitB)

Questions

- Thanks to:
  - Sander Vos
  - Steven Raspe
- Further questions:
  - alexandre.miguelferreira@os3.nl
  - ferreira.alexandremiguel@gmail.com