So Many Ways to Slap a YoHo: Hacking Facebook & YoVille
Its Medicinal!

I am an application security specialist (A.S.S.) who has been working in the field of Internet and application security for 10 years. These days I spend most of my time doing 'mad scientist' type security research. Currently deep in Shellcode.

Tom is growin' the good stuff in FarmVille! Tom attained the level of Hydro Farmer in FarmVille! He can now grow the OG Kush.

8 hours ago via FarmVille · Comment · Like · Play FarmVille now

Clarence Doskoci! The only FarmVille post I can appreciate. Good going man!!
27 minutes ago · Delete
What is YoVille?

- It’s part of the Zynga Family of games that are amazingly popular on Facebook.

YoVille Population: 5 Million Active Players
What is YoVille?

- Special thanks to Chris Peterson, VP of Application Security at Zynga.
- Zynga was aware of the security issues we brought to their attention.
- A security fix for the issues we will discuss is in place.
- The Facebook Application API can still be abused.
Roadmap

1) Introduction

2) Client-side trust attacks within Application APIs

3) Attack Patterns Against Social Network Gaming

4) Impact of the attacks using YoVille as an example

5) How to keep your software off the stage at DEF CON
Client-Side Trust

Gateway App

X-Application Trust

APP1

APP2

User
✓ Amplification: Attacker can use one compromised account to attack that users friends via social gaming.

✓ Deception: Phishers can create messages to lure users to click on malicious links or buttons, in some cases with the URI masked

✓ Easy to Exploit: By using a MITM proxy an attacker can create fake but legitimate looking prizes, gifts, or awards. Manipulating the API is trivial.
Attack Characteristics

- **Trust**: Since the attacks often originate from in-game friends or neighbors there is a greater tendency to trust the content as legitimate.

- **Stealth**: Because the attacks happen at the layer of application logic they are very difficult to detect (i.e. no noisy metacharacters or scripts).

- **Urgency**: Users are trained to quickly click and claim items their friends discover before the item expires or is used up.
1) Application API Manipulation via Man-in-the-Middle
2) Application API Content Spoofing via API Manipulation
3) Transaction or Event Tampering via API Manipulation
4) Transaction or Event Replay via API Manipulation
Attacks can prey on users interests or vanity

The best kind of lies are those we want to be believe...

Clicking Allow takes you to a Adobe PDF exploit

© Stracener, Tom 2010
Anatomy of a web 2.0 Application Framework

User
Client Browser

Master Application

User Page within Master Application is updated

JavaScript Java Flash

IFRAME CSS/XSL HTML

Web Application API

HTTP SSL REST API-KEY

JSON XML AJAX

Application Tier

Feeds/RSS Digital Signatures Encryption Persistent Storage

API-Request Broker

© Stracener, Tom 2010
Application Framework & API

- Friend Profile
- User Profile
- Event Feed
- Application Feed
- Application API
- Application Interface
- User Profile

© Stracener, Tom 2010
An attacker is able to modify message content or make API calls to create arbitrary content within cross-application Messages

- **Root Cause:** Failure to protect data from modification (i.e. failure to ensure data integrity).

- **Impact:** Attacker can create deceptive content that enables social engineering attacks, phishing, or user harassment
Content Spoofing Example

Content can be spoofed by modifying messages or creating new messages via direct query.
Attack Execution Flow: Spoofing

1. Direct API Call

2. Application Acknowledgement

3. Feed published to Gateway App via Feed

4. Gift or Message is Delivered to Target

5. Gift/Message Sent

Message can be a ‘wall update’ or a ‘gift’, for example.

© Stracener, Tom 2010
Cont.

Client

App API Method

Gateway App

App: GUI

Feed

GW: User News Feed

GW: Messages\Inbox

GW: Notification\Alerts

© Stracener, Tom 2010
5) Application API Navigation Remapping
6) Application API Button Hijacking
7) Harvesting Usernames via API Event Monitoring
8) Exploit Injection via Application API Message
9) Malware Propagation via Application API Message
Application Trust Boundaries

User

Client-Side Trust

Gateway App

APP1

APP2

Application Trust
When web application links that should point back to the application or its content are rewritten to trick users into following a malicious link.

- **Root Cause:** Failure to protect data from modification (i.e. failure to ensure data integrity).

- **Impact:** Potential compromise of user’s machine and/or accounts via direct exploitation of browser or plugin flaws. Potential for spoofing, phishing, & authorization of malicious applications.
API Request: Feed Processing

POST http://www.facebook.com/fbml/ajax/prompt_feed.php?__a=1 HTTP/1.1
Accept: */*
Accept-Language: en-us
x-svn-rev: 253302
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR 1.1.4322; .NET CLR 2.0.50727; .NET CLR 3.0.04506.31; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729; MS-RTC LM 8) Paros/3.2.13
Host: www.facebook.com
Content-length: 1661
Proxy-Connection: Keep-Alive
Pragma: no-cache
Cookie: datr=1254403372-89290de69f79ed40e6fde5039e27abf2e1179f524653:7168ae; lo=033Gi0x2EpsDBKf-I-HD2g; bx=strace%40gmail.com; bx=2; c_user=1409174187; bx=1; sctp=1274314001; sid=1; xs=87ea76b44fb16a3fb41794bee1182d; presence=DJ25933229G32hIL1409174187MF275933228563WMBicMsnDPBbIoMbvdMiJcMsbPBA_7zbQBAnUlBcncMsAOGBlAIDdAaST8V275933229Z40BlcPBcvrADrAO.9BtsP275932874QQG, force_hcb=1

MITM Proxy view
Link (Navigation) Tampering

- \&feed\_info[template\_data][name]=
  - Title of message,
  - clickable=Yes

- \&feed\_info[template\_data][href]=
  - URL for message title
  - URI masked=Yes

- \&feed\_info[template\_data][caption]=
  - Content of message
  - Clickable=No

- \&feed\_info[template\_data][media][0][src]=
  - Location of Image
  - URI masked=Yes
  - Clickable=No

- \&feed\_info[template\_data][media][0][href]=
  - Link for image within message
  - clickable=Yes
  - URI masked=Yes

- \&feed\_info[action\_link][0][text]=
  - Content of action text
  - URI masked=Yes
  - i.e. “Claim Mystery Keys”

- \&feed\_info[action\_link][0][href]=
  - Content of message
  - Clickable=No

- \&feed\_info[template\_data][media][0][src]=
  - Hyperlink for Message Image
  - URI masked=Yes

© Stracener, Tom 2010
Ass of Fire!
Lost and lonely hearts are Burning up in YoVille!

February 19 at 4:10pm via YoVille · Comment · Like · Collect the Ass of Fire
### Link Tampering

<table>
<thead>
<tr>
<th>Actor</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="media" alt="Image" />[src]= <img src="media" alt="Image" />[0][href]=</td>
</tr>
<tr>
<td>&amp;Feed_info[template_data][name]= [template_data][href]=</td>
</tr>
<tr>
<td>feed_info[template_data][caption]=</td>
</tr>
<tr>
<td>February 19 at 4:10pm via YoVille</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
Attack Execution Flow

1. Application Event

2. Application Event Message

3. Attacker Modifies Link Destinations

4. Event published to Gateway App via Feed

Game Application accessed via Gateway application

Game Application generates a structured API request to update Gateway App.
5) Application API Navigation Remapping

6) Application API Button Hijacking

7) Harvesting Usernames via API Event Monitoring

8) Exploit Injection via Application API Message

9) Malware Propagation via Application API Message
Application API Button Hijacking
CAPEC: 388

You’ll have to come see the talk. ;-)
Attack Patterns

5) Application API Navigation Remapping

6) Application API Button Hijacking

7) Harvesting Usernames via API Event Monitoring

8) Exploit Injection via Application API Message

9) Malware Propagation via Application API Message
Malware Propagation via Application API Message
CAPEC: 391

You’ll have to come see the talk. ;-)
How to Keep Your Software off the Stage at DEF CON

To build secure software you MUST understand how it will be attacked

A broad understanding of the attackers perspective resides in the heads of a relatively small group of people - most of them are here this weekend

The only way to scale this knowledge is to capture and share it in a structured and standardized way

http://capec.mitre.org
http://cwe.mitre.org

© Stracener, Tom 2010
# Attack Patterns for Social Gaming

<table>
<thead>
<tr>
<th>Attack Pattern</th>
<th>CAPEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application API Manipulation via Man-in-the-Middle</td>
<td>CAPEC-383</td>
</tr>
<tr>
<td>Application API Content Spoofing via API Manipulation</td>
<td>CAPEC-384</td>
</tr>
<tr>
<td>Transaction or Event Tampering via API Manipulation</td>
<td>CAPEC-385</td>
</tr>
<tr>
<td>Transaction or Event Replay via API Manipulation</td>
<td>CAPEC-386</td>
</tr>
<tr>
<td>Application API Navigation Remapping</td>
<td>CAPEC-387</td>
</tr>
<tr>
<td>Application API Button Hijacking</td>
<td>CAPEC-388</td>
</tr>
<tr>
<td>Harvesting Usernames via API Event Monitoring</td>
<td>CAPEC-389</td>
</tr>
<tr>
<td>Exploit Injection via Application API Message</td>
<td>CAPEC-390</td>
</tr>
<tr>
<td>Malware Propagation via Application API Message</td>
<td>CAPEC-391</td>
</tr>
</tbody>
</table>
Prescriptive Guidance
In other words: How do I avoid this $H17?

You’ll have to come see the talk. ;-)