Finding and Exploiting Access Control Vulnerabilities in Graphical User Interfaces

Collin Mulliner
crm[at]ccs.neu.edu

Black Hat USA, Las Vegas, August 2014
About

- Researcher at Northeastern University (Boston, MA)
  - Systems Security
  - Offense and Defense
  - Mobile

- This talk is based on the paper:
  **Hidden GEMs: Automated Discovery of Access Control Vulnerabilities in Graphical User Interfaces**
  *Collin Mulliner, William Robertson, Engin Kirda*
  35th IEEE Symposium Security and Privacy
  San Jose, CA, May 2014

- Materials for this talk will be available at:
  [http://mulliner.org/security/guisec](http://mulliner.org/security/guisec)
Graphical User Interfaces (GUIs)

- De facto standard to interact with most computing devices
  - Desktop, smart phone, computer-based appliances, ...

![Graphical User Interfaces](image-url)
Agenda

- GUI Security Background / History
- Basics of Graphical User Interfaces
- Access Control in the UI?!?!
- Introduction of GUI Element Misuse (GEMs)
- Automated app analysis to find GEM bugs
- Countermeasures
- Conclusions
GUI Security History (Shatter Attacks)

- Shatter Attacks

- Affected platform: Windows NT/2000/XP

- Remove limits of text edit fields
  - Paste input to cause memory corruption → code execution

- Target: progress with system privileges
  - Code execution → privilege escalation

- Now Windows has User Interface Privilege Isolation (UIPI)
  - Can't manipulate UI of process that have higher privileges
GUI Security History (Shatter Attacks)

- Shatter Attacks

- Affected platform: Windows NT/2000/XP

- Remote

- Target: progress with system privileges
  - Code execution → privilege escalation

- Now Windows has User Interface Privilege Isolation (UIPI)
  - Can't manipulate UI of process that have higher privileges

This talk is about Access Control issues in the UI
Graphical User Interfaces (GUIs)

- Windows, Widgets, ...
GUls → Widgets and Windows

- Widget → base UI element
  - Smallest element in a UI framework
  - On MS Windows: widget = window

- Common widgets
  - Window
  - Frame
  - Button
  - Check-box
  - Text edit field
  - Drop down box
  - Slider
Widget Attributes

- Attributes allow to change widget behavior at runtime
  - Allows user interface to be dynamic

- Common attributes

  Enabled → enable / disable widget

  Visibility → show / hide widget

  Read/Write → allow / disallow changing data stored in widget
**Widget Attributes**

- Attributes allow to change widget behavior at runtime
  - Allows user interface to be dynamic

- Common attributes
  - Enabled
  - Visibility
  - Read/Write

```
Login button disabled → indicates username required
```
Access Control

- Basic security requirement
- Common in any kind of enterprise application
- Especially applications that handle sensitive data
- Different privilege levels
  - Create / Add data
  - View data
  - Modify data
  - Execute privileged functionality
Access Control

- Basic security requirement
- Common in any kind of enterprise application
- Especially applications that handle sensitive data
- Different privilege levels
  - Create / Add data
  - View data
  - Modify data
  - Execute privileged functionality
- Implementing access control using the GUI is tempting
Access Control in the GUI

Disabled Button

Application Specific User
Access Control in the GUI

- Button used for access control
- Disabled Button
- Application Specific User
Access Control in the GUI

- Widgets can be manipulated
  - Feature of UI frameworks
  - No need to modify application binary

- Manipulate widget → bypass GUI-based access control
A Real World Attack **DEMO**
Access Control in the GUI

- Widgets can be manipulated
  - Feature of UI frameworks
  - No need to modify application binary

- Manipulate widget → bypass GUI-based access control

- Attacks using the UI are folklore

- First to systemantically investigate GUI security
Contributions

- We introduce **GUI Element Misuse (GEMs)**
  - Novel class of security vulnerabilities
  - Misuse of GUI elements for access control

- We define three classes of GEMs
  - Information Disclosure and Modification, Callback Execution

- Developed GEM Miner to automatically find GEMs
  - Find and verify GEMs in black box fashion

- We evaluated GEM Miner on applications for MS Windows
  - Found a number of GEMs in commercial software

- Releasing some tools today!
Threat Model

- Applications with internal user management
  - Multiple users or user and administrator
  - Accounts are NOT backed by the OS

- Accounts have different privileges
  - Reading vs. writing data
  - Executing privileged functionality

- Application domain
  - Enterprise applications → users with different privileges
  - Applications that manage data → require access control
GUI Element Misuse (GEM)

- Misusing GUI elements to implement access control

- GEM vulnerability → access control bypass vulnerability

- GEM classes
  - Unauthorized Callback Execution
  - Unauthorized Information Disclosure
  - Unauthorized Information Manipulation
Unauthorized Callback Execution

- Activation of UI element results in callback execution
  - Click button → execute callback → perform operation

- Assumption
  - Disabled UI element cannot be interacted with

- Attack
  - Enable UI element
  - Interact with UI element
  - Execute callback → perform operation
Unauthorized Callback Execution DEMO

- WinSpy++
  - http://www.catch22.net/software/winspy-17
  - They provide source, thanks!
Unauthorized Information Disclosure

- UI element is used to store sensitive information
  - UI element is shown only to privileged user

- Assumption
  - Hidden UI element cannot be made visible

- Attack
  - Set UI element visible
    - UI element is drawn by the UI framework
      - Data stored in UI element can be accessed
    - Access data stored in UI element programmatically
Unauthorized Information Disclosure DEMO

- `gemtools_unhide.exe`
  - Make all widgets of an application visible
  - Take screenshots of app windows
  - Tool is released today!
Dangling Information Disclosure

- Sensitive information is not scrubbed from UI element
  - Role-switch: user → privileged user → user

- Assumption
  - Hidden UI element cannot be made visible

- Attack
  - Set UI element visible
    - UI element is drawn by the UI framework
      - Data stored in UI element can be accessed
    - Access data stored in UI element programmatically
Unauthorized Data Modification

- **UI element is used to display and edit data**
  - Privileged user can edit data
  - Unprivileged user can view data

- **Assumptions**
  - Read-Only UI element does prevent data modification
  - Data modified only if element was writable → save data

- **Attack**
  - Set UI element Read-Write
    - Set/Change data
    - Click “save”
Unauthorized Data Modification DEMO

- WinSpy++ gemcolors edition!
  - Identify R/W settings of widgets
Widget Configuration

- User1 (Low Privileges)  
- User2 (High Privileges)
Applications must be executed by the same OS user
  - Interaction between apps via IPC

Attack steps:
  - Discover UI elements (widgets)
  - Obtain window HANDLE for widget
  - Manipulate widget
Technical

- All this is done through very basic Win32 APIs
  - SendMessage() family of functions
  - EnableWindow()
  - SendInput()
  - EnumChildWindows() → get all windows
  - SetWindowPos() → visible/hide window
  - GetWindowLong()
  - IsWindowEnabled()
  - IsWindowVisible()
  - GetClassName()

- This stuff is very well documented
UI Frameworks

- On MS Windows a window is the basic UI element
  - Everything is a window

- Win32 API provides basic functionality
  - 'actual' window
  - Button
  - Text field

- Other UI frameworks are build on top of the Win32 UI API
  - Provide their own widget types
  - Implement drawing and receiving user input
Win32 vs. .NET

- **.NET**
  - Win32 windows + custom widgets
  - Implement drawing and receiving user input
  - Win32 API can see widget but not always manipulate it

- **Attacker**
  - Can use Win32 API to interact .NET widgets
    - Enough for most attacks
  - Using .NET API provides access to actual .NET widgets
    - e.g., see individual buttons inside a 'button bar'

.NET 'button bar' for Win32 this is one button, for .NET it is 19
Two Corner Stones of GEM Vulnerabilities

- **False assumptions by developers**
  - GUI cannot be changed externally
    - Widget attributes are protected

- **Non sophisticated attacker**
  - Only point-and-click
  - Black box attack → change value in field OR click button
    - No reverse engineering or program understanding
    - Don't need to manually temper with files or database
    - No network protocol knowledge
GEM Attacks

- Easy to carry out
  - Anybody can do it (if they know how to use a computer)

- Fast
  - Are you still trying to find the location of the binary?
The GEM Miner Analysis

- Systematically test applications for GEM vulnerabilities
  - Automated analysis
  - Complex applications cannot be tested manually

- Black box analysis
  - We do NOT require: source code, reverse engineering, etc.
The GEM Miner System

- Explore application UI and record widgets and attributes
- Identify GEM candidate widgets
- Check the GEM candidates
Application Seeding

- Create application specific users
  - Users + administrator

- Create data
  - e.g., items of an inventory management system

- Configure access control (restrict privileges of one account)
UI Exploration

- Explore the application's UI
  - Interact with widgets
    - click button, set check box, select drop down, ...

- Record
  - Widgets and attributes
  - Interactions
UI Exploration – for all privilege levels

- UI Exploration is executed once for each distinct privilege level
- Result: UI State for each privilege level
- UI State
  - Windows, contained widgets, and their attributes
GEM Candidate Identification

- Compare UI States of different privilege levels
  - Widget with different attributes → GEM candidate

<table>
<thead>
<tr>
<th>Level</th>
<th>Attributes</th>
<th>UI Element</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>Visible Disabled</td>
<td>TbitBtn</td>
<td>“New Article”</td>
</tr>
<tr>
<td>High</td>
<td>Visible Enabled</td>
<td>TbitBtn</td>
<td>“New Article”</td>
</tr>
<tr>
<td>Low</td>
<td>Visible Enabled</td>
<td>TbitBtn</td>
<td>“Help”</td>
</tr>
<tr>
<td>High</td>
<td>Visible Enabled</td>
<td>TbitBtn</td>
<td>“Help”</td>
</tr>
<tr>
<td>Low</td>
<td>Visible Enabled Read</td>
<td>EDIT</td>
<td>“”</td>
</tr>
<tr>
<td>High</td>
<td>Visible Enabled Write</td>
<td>EDIT</td>
<td>“”</td>
</tr>
</tbody>
</table>
GEM Candidates

- **GEM Candidate**
  - Widget that likely can be used to bypass access control

- **Candidate information**
  - Widget type and ID
  - Path to candidate widget
  - “successor” (e.g. if widget creates a new window)
GEM Checking

- Execute AUT
- Drive application to GEM candidate
- Test GEM candidate
  - Manipulate and activate widget
  - Inspect result
GEM Candidate Testing

- Different strategy for each widget and GEM type
  - Callback execution: active widget → callback executed?
  - Information disclosure: can widget contain data?
  - Information modification: modified data accepted by app?

- Black box testing
  - Manipulate the UI for testing
  - Check results by only inspecting the UI

- Tests are independent from the application
  - No application specific knowledge needed
Testing Callback Widgets

- What effect does 'activation' of widget have?
  - e.g. new window / popup?
Testing for Information Disclosure

- No actual testing required

- Conditions
  - Widget is not visible in “low privileged” mode
  - Widget can store data
Testing Data Modification GEMs 1/4

- Drive application to window containing GEM candidate
Testing Data Modification GEMs 2/4

- Set text edit field writable
- Change/Set test value
- Close window
Testing Data Modification GEMs 3/4

- Drive application to window containing GEM candidate
- Check if test value present
Testing Data Modification GEMs 4/4

- Drive application to window containing GEM candidate
- Check if test value present

GEM Candidate confirmed!
Result → GEMs no longer hidden!

GEM Analysis

AUT

UI Exploration → GEM Candidate Identification → GEM Checker

UI States

GEM Candidates

Test Engineer

Widget + Type
Window
Path to Widget

GEMs

“Hidden GEMs”
### Analyzing Real World Apps (Evaluation)

<table>
<thead>
<tr>
<th>Application</th>
<th>GEM Candidates</th>
<th>Automatically Confirmed</th>
<th>Manually Confirmed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Disclosure Mod Modification Callbacks</td>
<td>Disclosure Mod Modification Callbacks</td>
<td>Mod Modification Callbacks Runtime</td>
</tr>
<tr>
<td>App1</td>
<td>44 - 2</td>
<td>44 - 2</td>
<td>- - 51 sec</td>
</tr>
<tr>
<td>App2</td>
<td>1 1 8</td>
<td>- - 4</td>
<td>- 2 205 sec</td>
</tr>
<tr>
<td>Proffix</td>
<td>- 23 10</td>
<td>- 17 7</td>
<td>3 1 666 sec</td>
</tr>
<tr>
<td>Total</td>
<td>45 24 20</td>
<td>44 17 13</td>
<td>3 3</td>
</tr>
</tbody>
</table>

- **App1**: inventory management
  - Multiple users + admin mode

- **App2**: employee and project management
  - Multiple users + admin

- **Proffix**: customer relationship management
  - Multiple users + admin, fine-grained access control
Analyzing Real World Apps (Evaluation)

<table>
<thead>
<tr>
<th>Application</th>
<th>GEM Candidates</th>
<th>Automatically Confirmed</th>
<th>Manually Confirmed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Disclosure</td>
<td>Modification</td>
<td>Callbacks</td>
</tr>
<tr>
<td>App1</td>
<td>44</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>App2</td>
<td>1</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>Proffix</td>
<td>-</td>
<td>23</td>
<td>10</td>
</tr>
<tr>
<td>Total</td>
<td>45</td>
<td>24</td>
<td>20</td>
</tr>
</tbody>
</table>

- App1: inventory management
  - Multiple users + admin mode

- App2: employee and project management
  - Multiple users + admin

- Proffix: customer relationship management
  - Multiple users + admin, fine-grained access control

Win32

.NET
## Results – Callback GEMs

<table>
<thead>
<tr>
<th>Application</th>
<th>GEM Candidates</th>
<th>Automatically Confirmed</th>
<th>Manually Confirmed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Disclosure</td>
<td>Modification</td>
<td>Callbacks</td>
</tr>
<tr>
<td>App1</td>
<td>44</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>App2</td>
<td>1</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>Profix</td>
<td>-</td>
<td>23</td>
<td>10</td>
</tr>
<tr>
<td>Total</td>
<td>45</td>
<td>24</td>
<td>20</td>
</tr>
</tbody>
</table>

- App2: disables button to deny export DB functionality
  - Enable button → execute export DB

- Unconfirmed candidates
  - Actual access control
Results – Information Disclosure GEMs

<table>
<thead>
<tr>
<th>Application</th>
<th>GEM Candidates</th>
<th>Automatically Confirmed</th>
<th>Manually Confirmed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Disclosure</td>
<td>Modification</td>
<td>Callbacks</td>
</tr>
<tr>
<td>App1</td>
<td>44</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>App2</td>
<td>1</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>Proffix</td>
<td>-</td>
<td>23</td>
<td>10</td>
</tr>
<tr>
<td>Total</td>
<td>45</td>
<td>24</td>
<td>20</td>
</tr>
</tbody>
</table>

- **App1**: creates a large number of top-level windows on startup
  - Including the user management window

- **App1**: dangling disclosure
  - Switch: user → admin → user
  - Admin password in hidden window
Results – Information Modification GEMs

<table>
<thead>
<tr>
<th>Application</th>
<th>GEM Candidates</th>
<th>Automatically Confirmed</th>
<th>Manually Confirmed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Disclosure</td>
<td>Modification</td>
<td>Callbacks</td>
</tr>
<tr>
<td>App1</td>
<td>44</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>App2</td>
<td>1</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>Proffix</td>
<td>-</td>
<td>23</td>
<td>10</td>
</tr>
<tr>
<td>Total</td>
<td>45</td>
<td>24</td>
<td>20</td>
</tr>
</tbody>
</table>

- **Proffix**: R/W access control for database via text field attribute
  - Red boxes → Read-Only text fields

- **Unconfirmed candidates**
  - Field cannot be changed
  - Field relies on other value
Summary

- **GEM Vulnerabilities**
  - Exist in commercial software
  - Can be exploited by non sophisticated attackers

- **GEM Miner Analysis**
  - Systematic method to find GEM vulnerabilities
  - Independent of UI framework and application

- **The GEM Miner System**
  - Can automatically find and verify GEM bugs
  - Implemented for Windows but can be ported to other OSes
Other OSes and GUI Toolkits

- GEM bugs can be exploited if:
  - GUI of application can be inspected
  - GUI elements can be manipulated

- Proof-of-Concept for GTK+ on Linux
  - (just because it is totally different)
  - LD_PRELOAD a library into GTK+ application
    - Discover widgets
    - Modify widget attributes
Countermeasures

- Application developers should not rely on the GUI framework
  - Don't store runtime information in UI elements
  - Treat data stored in widgets as untrusted user input
  - Create and destroy widgets and windows as needed

- Remove unused UI elements from the UI
  - Can't manipulate non-existent elements
  - “Partial fix only”

- Run vulnerable application as different OS user
  - This will prevent manipulating the UI
  - This is an easy to deploy HOT FIX
Conclusions

- We introduced GUI Element Misuse (GEMs)
  - New class of security vulnerabilities
  - Misuse of the UI to implement access control

- We defined three classes of GEMs
  - Information Disclosure and Modification, Callback Execution

- We build GEM Miner to analyze Windows applications for GEMs
  - We discovered a number of previously-unknown bugs

- First step towards including the UI in security testing
  - We specifically address access control vulnerabilities
Thank you!

Any Questions?

http://mulliner.org/security/guisec/
Future Work

- Look at more applications!
- Appliances that run custom UI apps on standard OSes
- Detailed investigation of other OSes and GUI frameworks