Northeastern University
Systems Security Lab

Finding and Exploiting Access Control Vulnerabilities in Graphical User Interfaces

Black Hat USA 2014

Collin Mulliner
crm[at]ccs.neu.edu

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

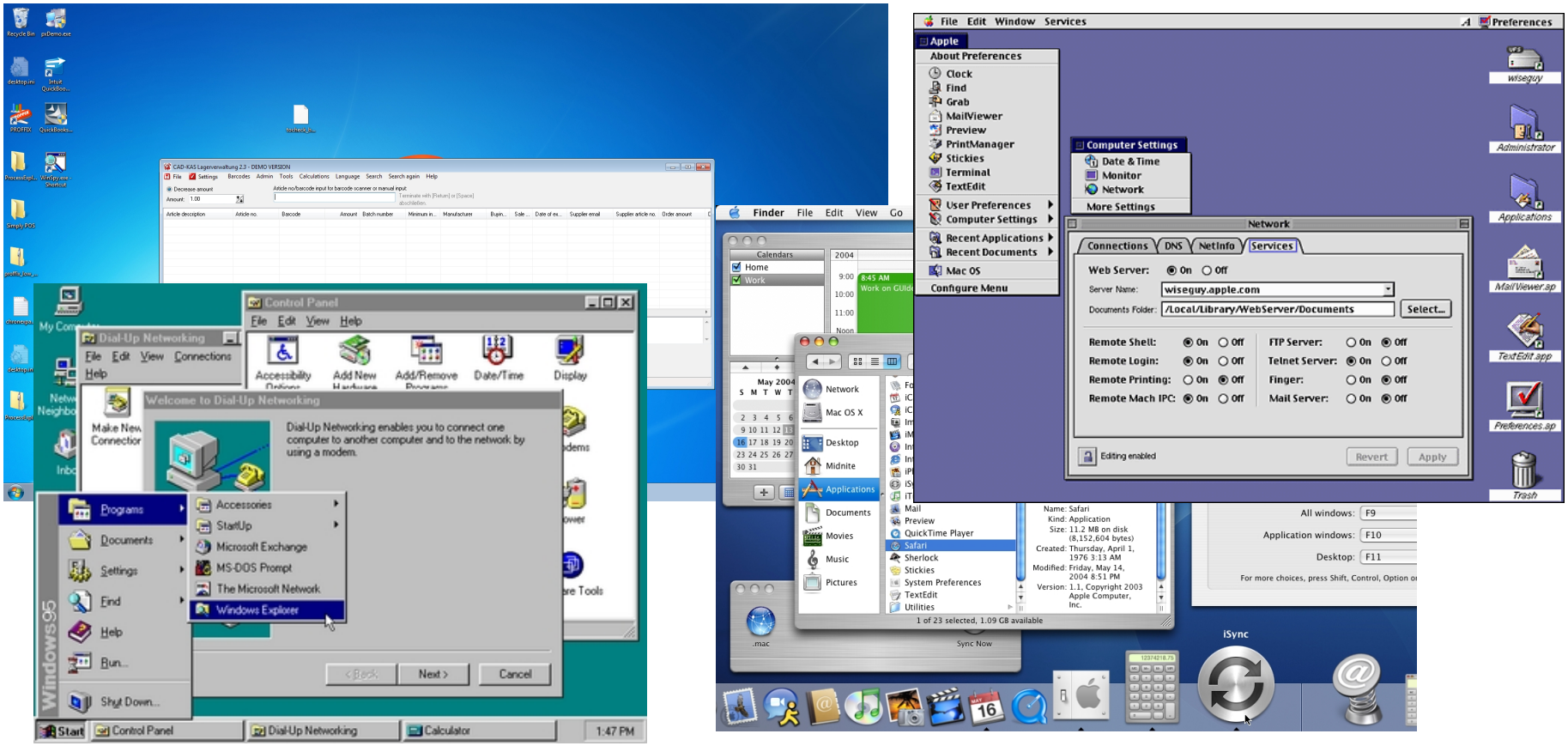
  San Jose, CA, May 2014

- Materials for this talk will be available at:

  http://mulliner.org/security/guisec

Graphical User Interfaces (GUIs)

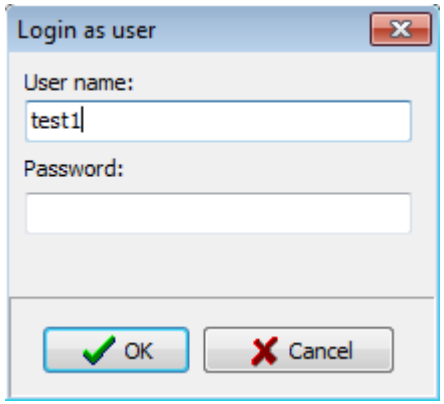
- De facto standard to interact with most computing devices
  - Desktop, smart phone, computer-based appliances, ...

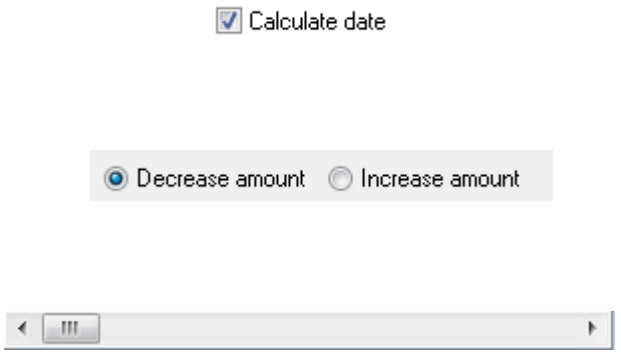


GUIs → Widgets and Windows

- 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 attribut
  - Enabled
  - Visibility
  - Read/Write

a stored in widget

Login button disabled → indicates username required

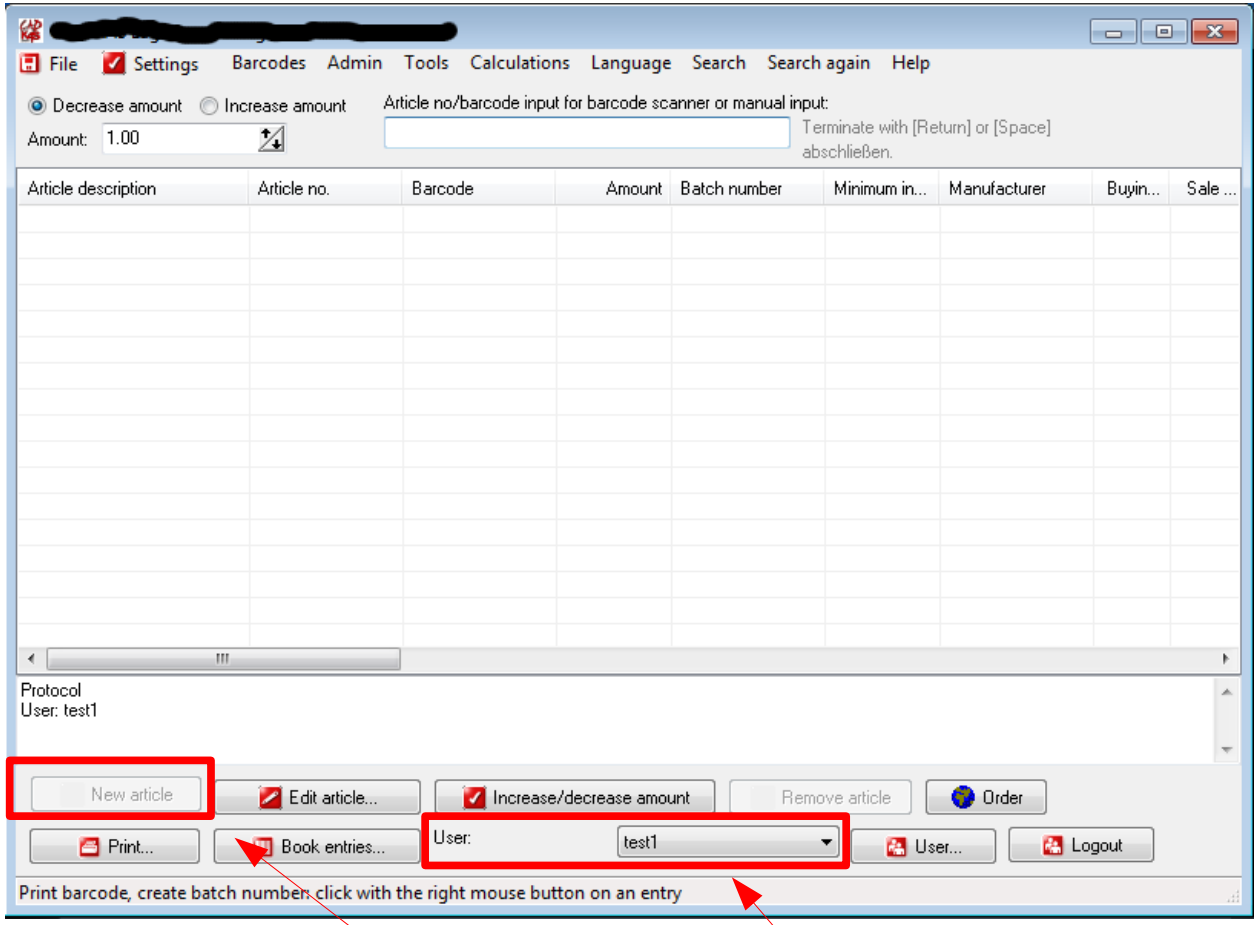
Access Control

- 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

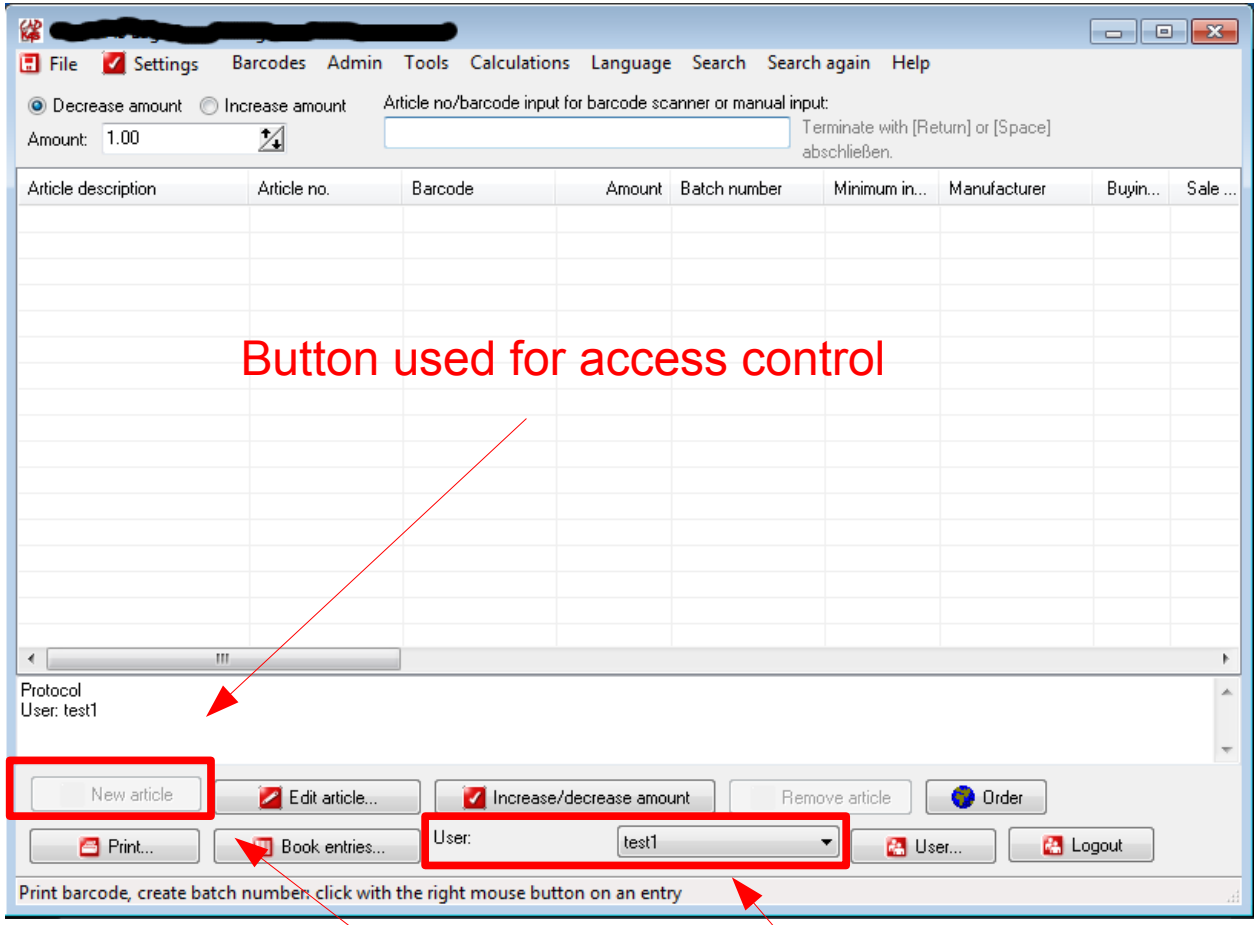
Access Control

- 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



Access Control in the GUI



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

- **We are 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

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 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

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”

Two Corner Stones of GEM Vulnerabilities

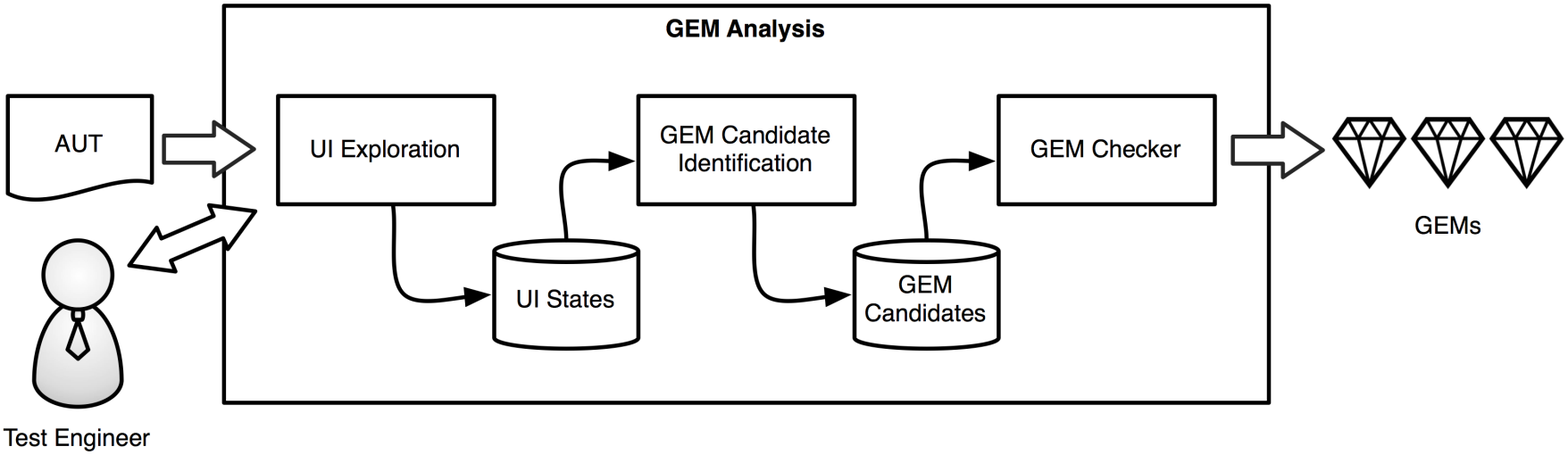
- **False assumptions by developers**
  - GUI cannot be changed externally
    - Widget attributes are protected

- **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

The GEM Miner Analysis

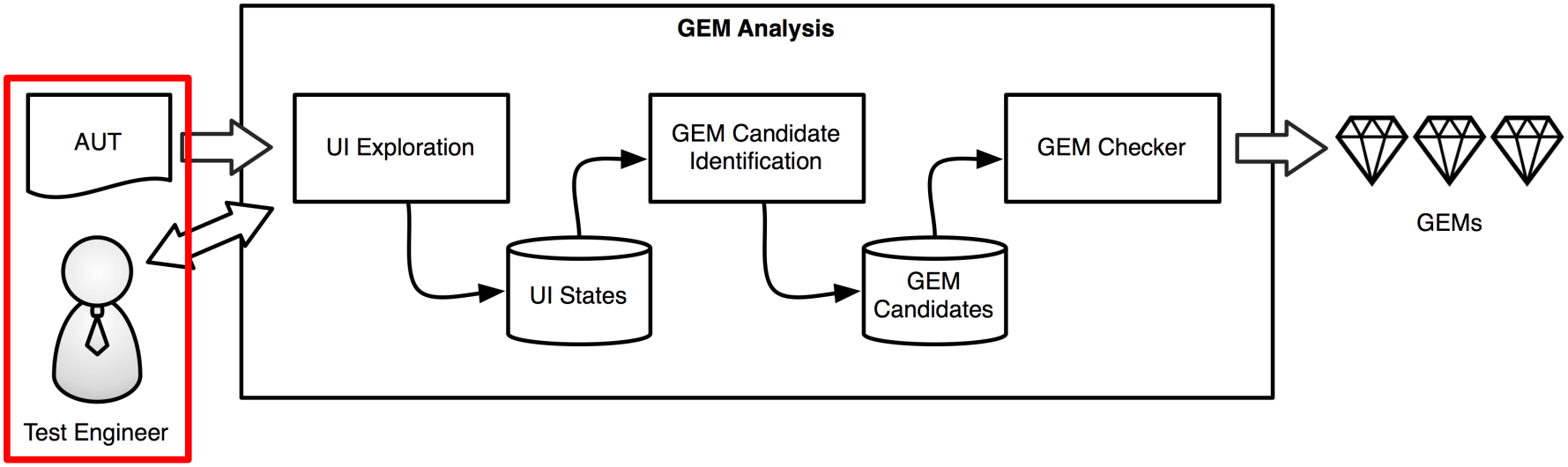
- Systematically test applications for GEM vulnerabilities
  - Automated analysis
  - Complex applications cannot be tested manually
- Black box analysis
  - We do NOT require: source code, reverse engineering, etc.

The GEM Miner System



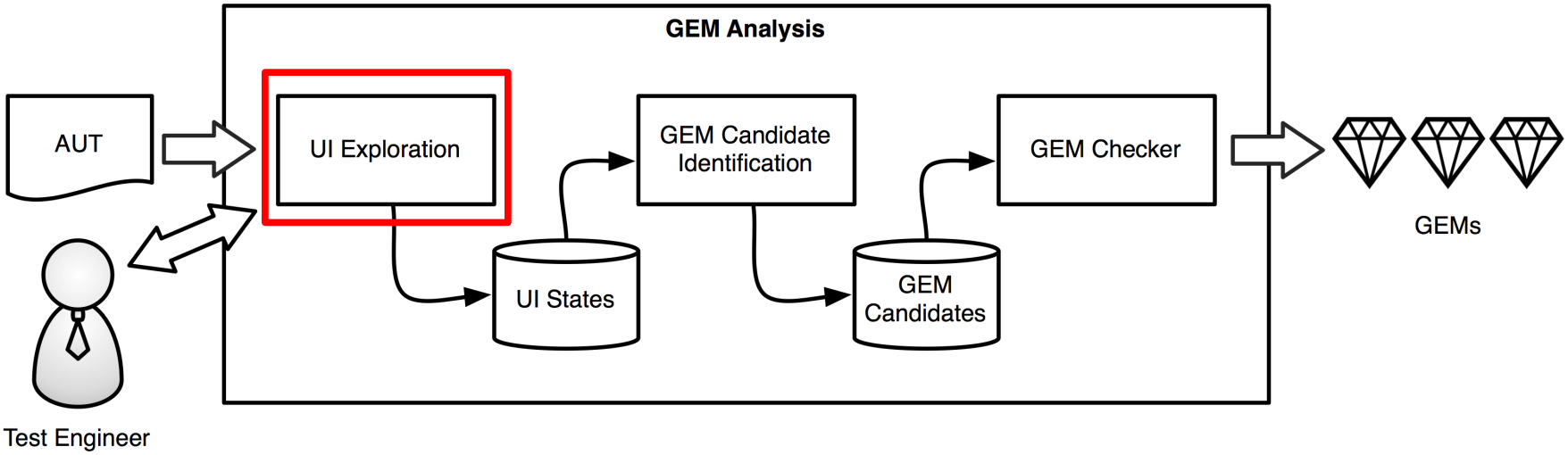
- Explore application UI and record widgets and attributes
- Identify GEM candidate widgets
- Check the GEM candidates

Application Seeding



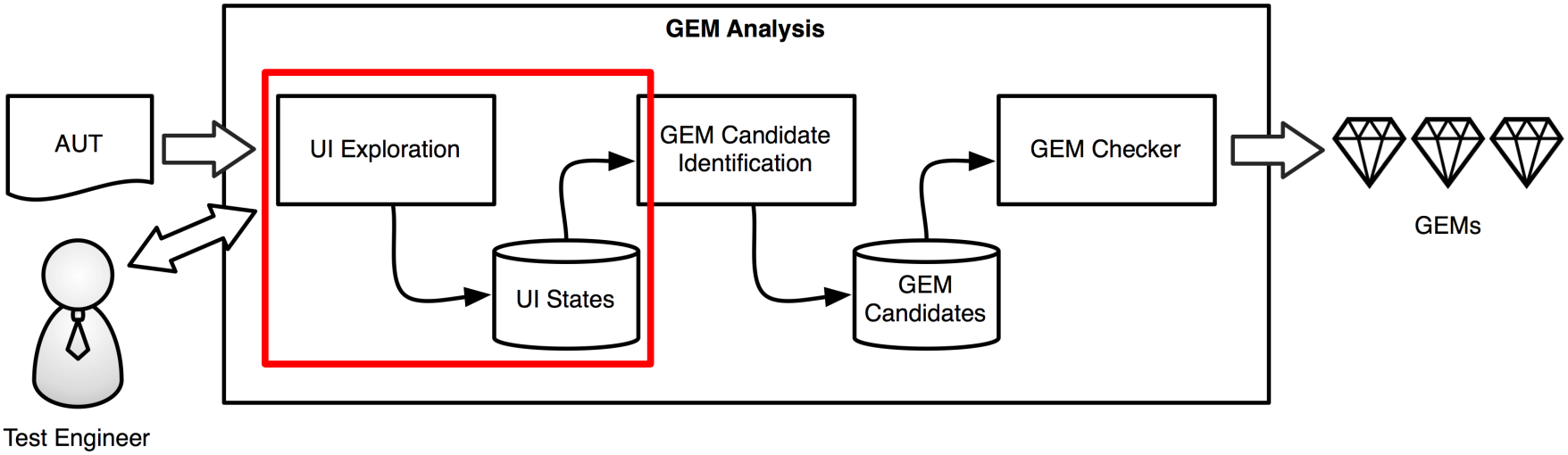
- Create application specific users
  - Users + administrator
- Create data
  - e.g., items of an inventory management system
- Configure access control (restrict privileges of one account)

UI Exploration



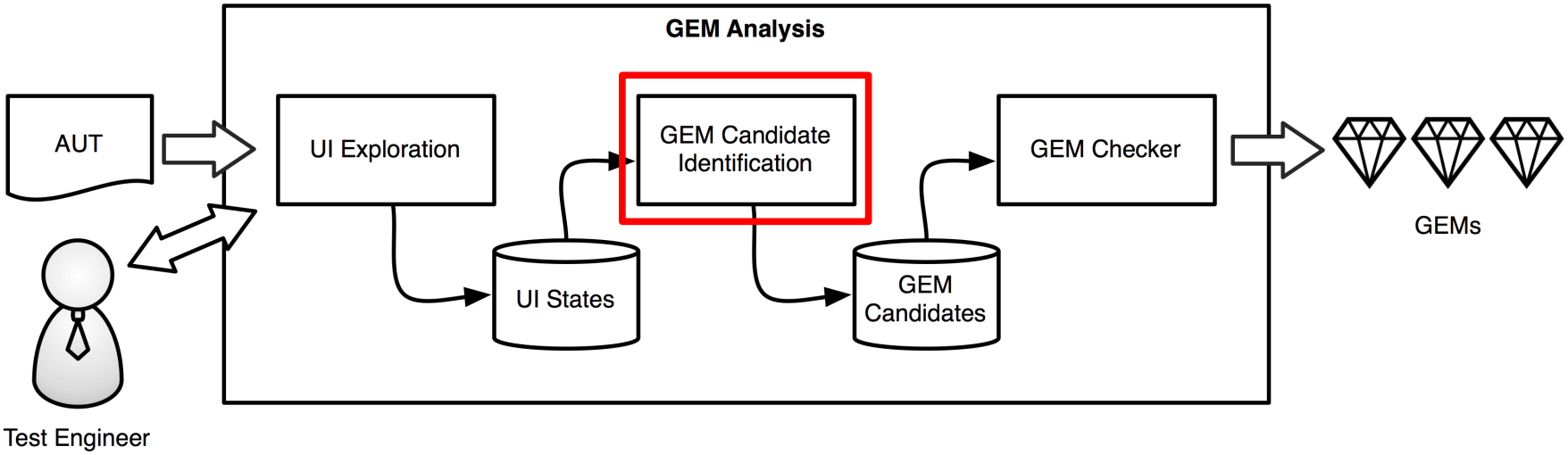
- Explore the application's UI
  - Interact with widgets
    - click button, set check box, select drop down, ...
- Record
  - Widgets and attributes
  - Interactions

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

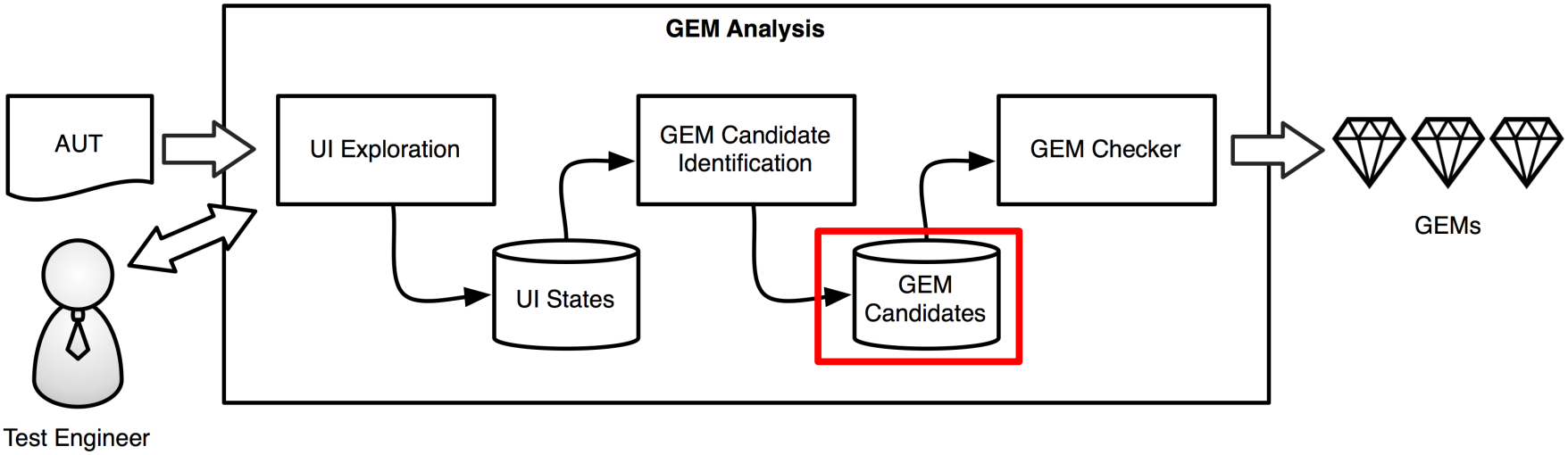
GEM Candidate Identification



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

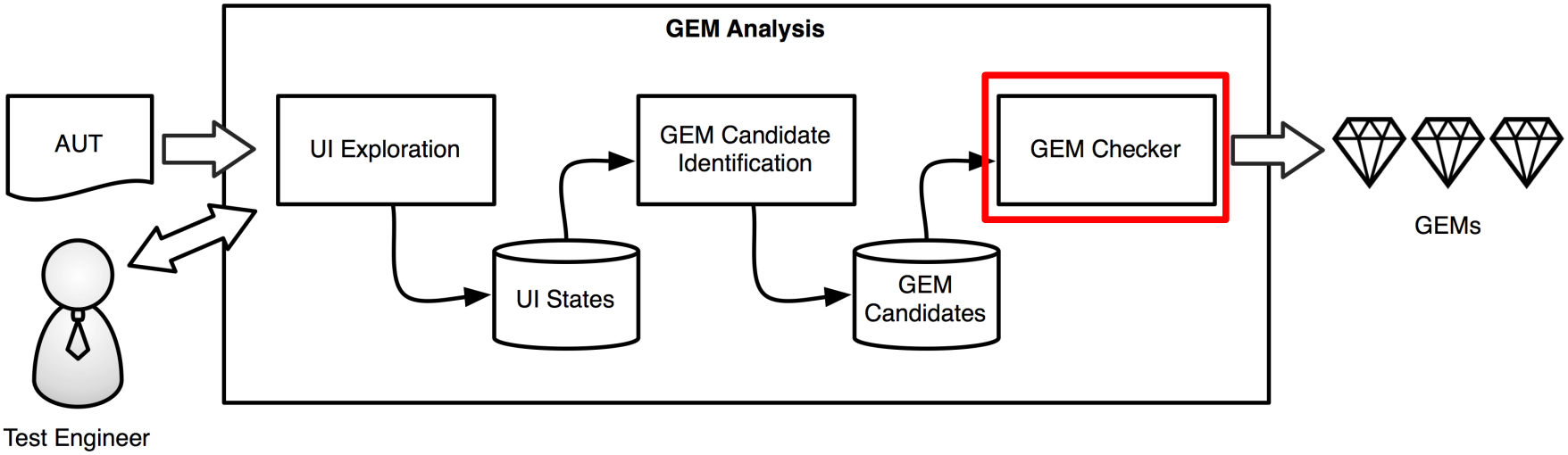
Level	Attributes	UI Element	Label
Low	Visible Disabled	TbitBtn	"New Article"
High	Visible Enabled	TbitBtn	"New Article"
Low	Visible Enabled	TbitBtn	"Help"
High	Visible Enabled	TbitBtn	"Help"
Low	Visible Enabled Read	EDIT	""
High	Visible Enabled Write	EDIT	""

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

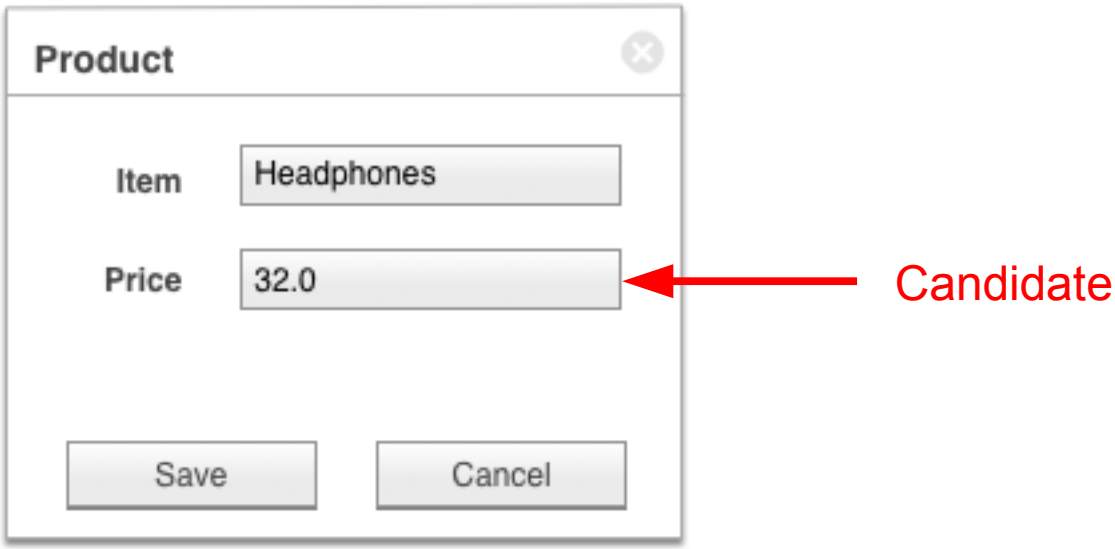
- Different strategy for each widget and GEM type
  - Callback execution: active widget → callback executed?
  - Information disclosure: can widget contain data?
  - Information modification: modified data accepted by app?

- 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

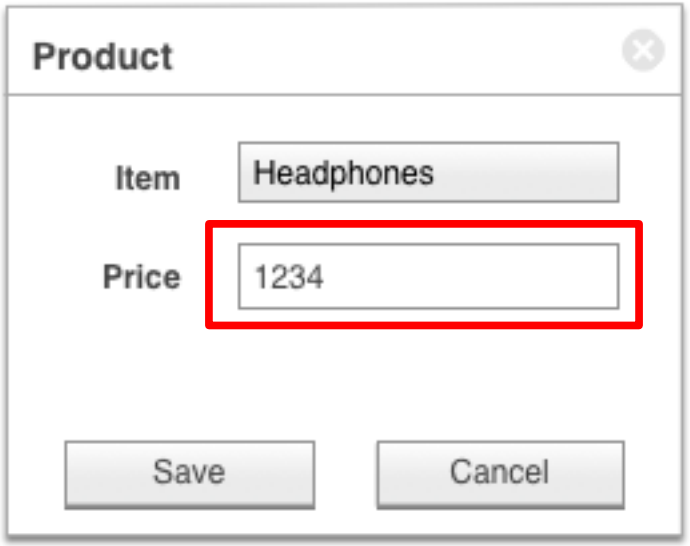
Testing Data Modification GEMs

- Drive application to window containing GEM candidate



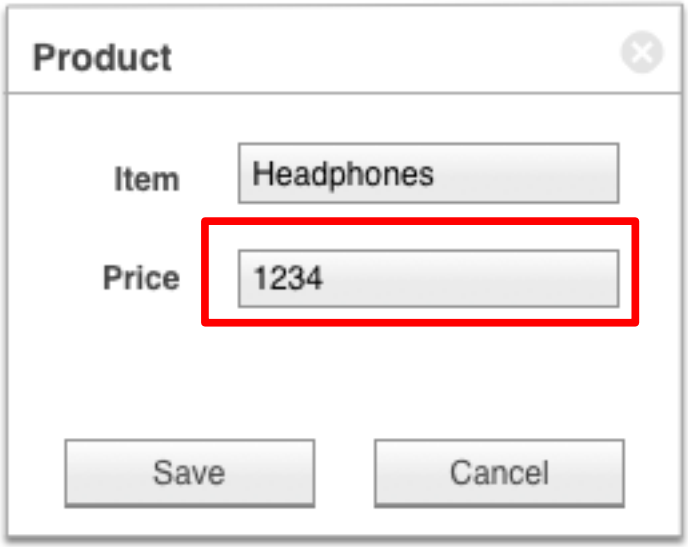
Testing Data Modification GEMs

- Set text edit field writable
- Change/Set test value
- Close window



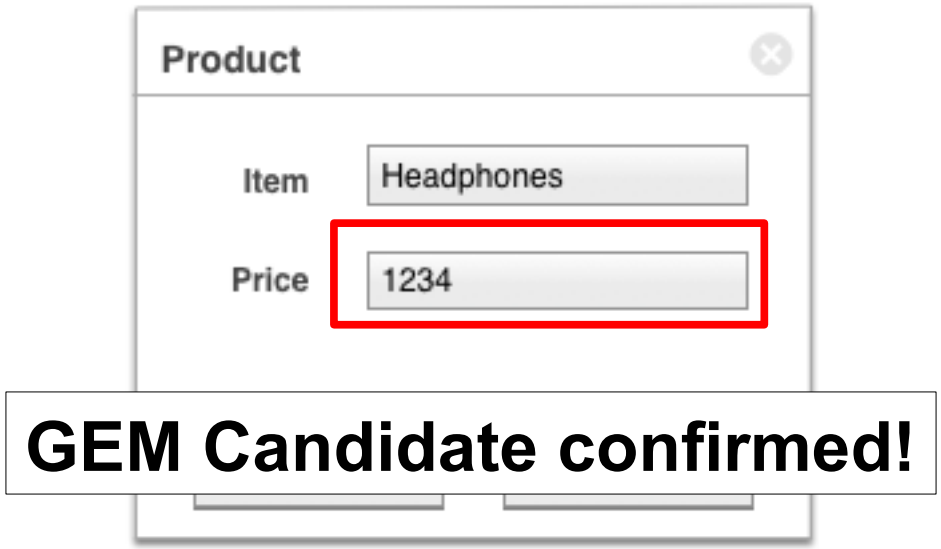
Testing Data Modification GEMs

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

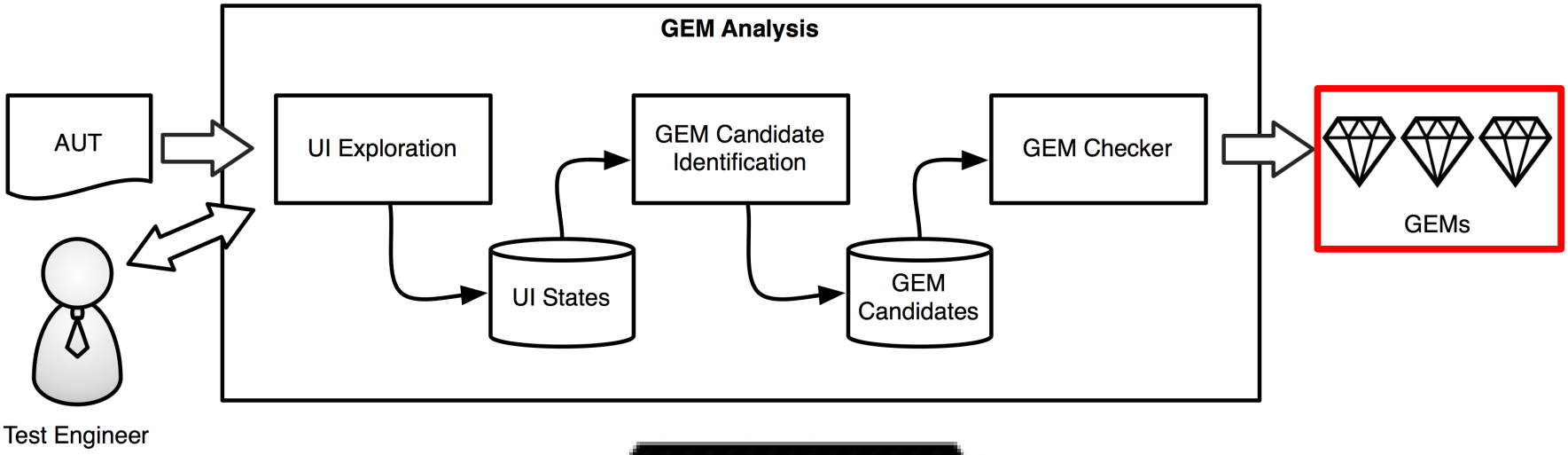


Testing Data Modification GEMs

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



Result → GEMs no longer hidden!



Evaluation

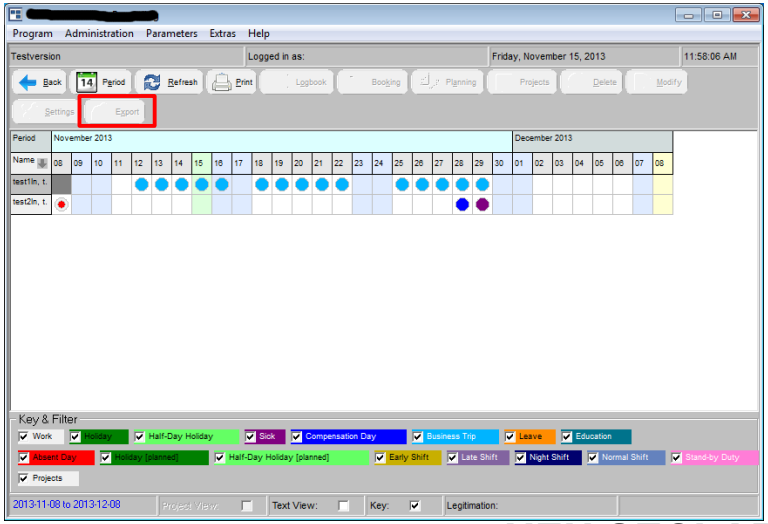
	GEM Candidates			Automatically Confirmed			Manually Confirmed		
Application	Disclosure	Modification	Callbacks	Disclosure	Modification	Callbacks	Modification	Callbacks	Runtime
App1	44	-	2	44	-	2	-	-	51 sec
App2	1	1	8	-	-	4	-	2	205 sec
Proffix	-	23	10	-	17	7	3	1	666 sec
Total	45	24	20	44	17	13	3	3	

- 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

Results – Callback GEMs

	GEM Candidates			Automatically Confirmed			Manually Confirmed		
Application	**Disclosure**	**Modification**	**Callbacks**	**Disclosure**	**Modification**	**Callbacks**	**Modification**	**Callbacks**	**Runtime**
App1	44	-	2	44	-	2	-	-	51 sec
App2	1	1	8	-	-	4	-	2	205 sec
Proffix	-	23	10	-	17	7	3	1	666 sec
Total	45	24	20	44	17	13	3	3	

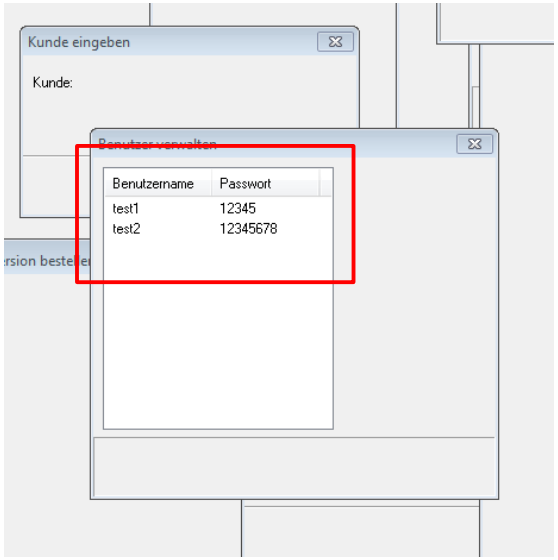
- App2 : disables button to deny export DB functionality
  - Enable button → execute export DB
- Unconfirmed candidates
  - Actual access control



Results – Information Disclosure GEMs

	GEM Candidates			Automatically Confirmed			Manually Confirmed		
Application	**Disclosure**	**Modification**	**Callbacks**	**Disclosure**	**Modification**	**Callbacks**	**Modification**	**Callbacks**	**Runtime**
App1	44	-	2	44	-	2	-	-	51 sec
App2	1	1	8	-	-	4	-	2	205 sec
Proffix	-	23	10	-	17	7	3	1	666 sec
Total	45	24	20	44	17	13	3	3	

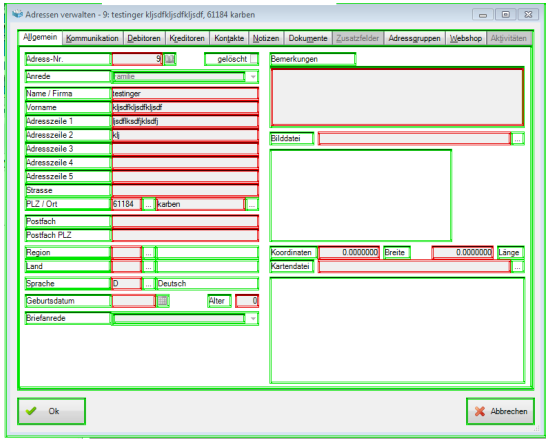
- App1: creates a large number of top-level windows on startup
  - Including the user management window
- App1: dangling disclosure
  - Switch: user → admin → user
    admin password in hidden window



Results – Information Modification GEMs

	GEM Candidates			Automatically Confirmed			Manually Confirmed		
Application	**Disclosure**	**Modification**	**Callbacks**	**Disclosure**	**Modification**	**Callbacks**	**Modification**	**Callbacks**	**Runtime**
App1	44	-	2	44	-	2	-	-	51 sec
App2	1	1	8	-	-	4	-	2	205 sec
Proffix	-	23	10	-	17	7	3	1	666 sec
Total	45	24	20	44	17	13	3	3	

- 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

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

EOF

Thank you!

Questions?