**Thwarting Malware and UI Redressing Attacks with Verifiable User Actions**

Jeff Shirley and David Evans
University of Virginia
Department of Computer Science

---

**Security is all about User Intentions**

A computer is secure if you can depend on it and its software to behave as you expect.

(Garfinkel, Spafford & Schwartz)

---

**Examples**

Malware  Phishing  CAPTCHAs

Next: Clickjacking

---

**Clickjacking (UI Redressing)**

http://www.cs.virginia.edu/evans/clickjack/demo.html

---

```
function onclick_event_handler(e) {
  var c = get_position(e); // returns position of mouse click
  document.getElementById('victim').setAttribute('style',
    'position:absolute;top:' + (c.y - 240) + ';left:' + (c.x - 840) + ';'); // move to button location
}
```
UI Redressing Claims

- No good server-side defense
  - Server sees two perfectly normal requests
- Client-side defenses
  - Change browser to prevent attack page
    - e.g., no transparent frames, better display-sharing policy
  - Need to break backwards compatibility
  - NoScript’s approach: warn when clicks reach hidden elements
- General defense: only allow actions that are consistent with user intentions

Our Goal

- Systematically incorporate user intentions in security policies

Outline:
- Securely capture user actions
- Robustly infer user intentions from those actions
- Express and enforce policies that incorporate user intentions

Related Work: Using User Intentions

SpoofGuard
[Chou, Ledesma, Teraguchi, Boneh, Mitchell, NDSS 2004]

“BINDER exploits a unique characteristic of personal computers, that most network activities are directly or indirectly triggered by user input.”

Polaris
[Cui, Katz & Tan, USENIX Tech 2005]

Not-a-Bot
[Gummadi, Balakrishnan, Maniatis, Ratnasamy, NSDI 2009]

How can we securely capture user intentions?

Capturing User Actions

VM Captures User Actions before they reach Guest OS

User-Intent Based Access Control
Main Challenges

- **Inferring User Intent**: depends on what user does and sees
- **Designing UIBAC Policies**: grant permissions based on the history of user intentions and program actions
- **Intercepting actions, enforcing policies**

Protecting User Interfaces

Intent of user action depends on apparent UI element user is interacting with

Visual Templates

Prototype compares visual output from a virtual machine to "visual templates" that specify look of user interface elements

Template Matching

- Templates consist of a bitmap image plus a set of regions that are ignored during comparison
  - Compare screenshot with image template
  - Use precomputed SHA-1 hash for speed
- Ignored regions generalize visual templates
  - Tradeoff between generality and exactness of UI matching

Template Challenges

Learning Templates

Collect screenshots of related dialogs by running trusted applications
Generalizing Templates

Generalize by clustering mismatched pixels, find minimal bounding boxes of varying regions, exclude from template

Inferring Intentions

- Many ways to express same intention
  - Mouse click sequences, keyboard shortcuts, etc.
  - Sets of rules of inferring particular abstract intentions
    - e.g., intent to open file f

User-Intent Based Policies

Policies

- Universal Policies
  - Anti-Malware: apply to all processes
    - access files selected by user (e.g., File open/save dialog)
    - access files and directories installed with application
    - access files created by application
  - Take advantage of user intentions to enable default strict policy that is relaxed based on user actions
- Application-Specific Policies
  - Resource: amazon.com/add-to-cart
  - Granted by: user click on template

Anti-Malware Policy

Mandatory Access Control
- Default deny
- Permissions granted based on history of all user interactions

Malware Preliminary Results

Prevention: 30 effective malware samples: all malicious behaviors prevented (except for limits in intercepting actions)

<table>
<thead>
<tr>
<th>False Positives</th>
<th>Program</th>
<th>Infection</th>
<th>Policy Violations</th>
<th>Dialog Validated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firefox 3.0.5</td>
<td>4739</td>
<td>2</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>iTunes 8.0</td>
<td>13382</td>
<td>0</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Windows Media Player 11.0</td>
<td>8217</td>
<td>8</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Wordpad 5.1</td>
<td>2887</td>
<td>0</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Word 2007</td>
<td>8303</td>
<td>5</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

Flash component files
Uses if as embedded component, accesses history and cookies

Uses non-standard UI elements
So, what about those CAPTCHAs?

**Externally-Verifiable User Actions**

- Network messages include TPM-signed tokens
  - **Option 1:** attest to filter that collects and signs screenshots and inputs
  - **Option 2:** attest to analyzer that signs verified events
- Sample Applications:
  - No more CAPTCHAs!
  - Eliminate click fraud (only pay for signed clicks?)
  - Prevent worm propagation
  - Non-repudiatable transactions

**Summary**

- Security is all about user intentions
  - Expressed through normal interactions, not security dialogs
  - Understanding them is hard: interpreting intentions depends on understanding what users do and see
- Lots of opportunities to use collected user intentions
  - Desktop User-Intent Based Access Control
  - Universal policies to thwart malware
  - Application-specific policies
  - Externally-verifiable user actions
  - Verifiable, non-repudiatable user transactions

---

**Shameless Book Plug**

http://www.computingbook.org/

1. Defining Procedures
2. Analyzing Procedures
3. Improving Expressiveness
4. Limits of Computing
5. Programming the Web

Main underlying themes:
- Recursive definitions
- Abstraction
- Universality (Programs/Data)