Abusing HTML5 permissions on browsers

Joel Niklaus

04/DEC/2018

SCG Software Composition Seminar
Hey - we're giving away iPad minis!!! Just click the WIN button and it's yours!!!

>> WIN <<
What would you like to do with all your money?

Invest it wisely

Donate it to Kim Dotcom
Hey - we're giving away iPad minis!!! Just click the WIN button and it's yours!!!

What would you like to do with all your money?

Invest it wisely

>>WIN<<

Do as you wish Kim Dotcom
Relevance

HTML5 browsers available on all major platforms

HTML5 supports numerous sensitive data providers
  (geolocation, camera, microphone, etc.)

# potential victims: billions
Motivation

**Exploration** of technical possibilities

In-depth **victim assessment**
Threats & Attacks
Threats

Risks
- Revelation of confidential information
- Hostile computer takeover
- Unintended participations of any kind

Gains
- Blackmailing as business model
- Increasing Twitter reach
- Influencing Facebook community
- Generating Google AdSense revenue
- Gaining popularity on YouTube

Spoofing:
- Display Pointer
- Temporal
A) Display Spoofing
B) Pointer Spoofing
C) Temporal Spoofing
Prevalence of Vulnerabilities

Google Explorer API
Facebook iframe
LinkedIn AutoFill plug-in
Defense Mechanisms

Frame busting

HTTP header X-Frame options

- **DENY**  any embedding disallowed
- **SAMEORIGIN**  only on the same website allowed
- **ALLOW FROM**  allowed on the specified websites

JavaScript (defense script / no-script add-on)

GuardedID
Technical Details
// Check for Geolocation API permissions
navigator.permissions.query({name:'geolocation'})
.then(function(permissionStatus) {
    console.log('geolocation permission state is ', permissionStatus.state);
    permissionStatus.onchange = function() {
        console.log('geolocation permission state has changed to ', this.state);
    };
});
Concept

Intention:  Trick users on browser permission dialog

Procedure:  1) Accurate positioning of a fake button
         2) Make the user click very fast
         3) Speed dependent permission dialog box trigger
         4) Distraction of the user
         5) Fake end screen
Experiments
Demo #01
Controlled Experiment

People with diverse backgrounds

Controlled environment
(same computer and browser)

The majority of people could be fooled!

% of people fooled
- # tricked:
  - 4%
- # not tricked:
  - 96%

% of people seeing dialog
- # noticed:
  - 16%
- # not noticed:
  - 84%
Uncontrolled Experiment

Computer Science Students

Different computers and browsers

Even people with a strong CS background could be fooled!
VIRUS
Infected System

• Permissions are stored in an unprotected SQLite database in the case of Firefox and Chrome on Windows

• Suppose a virus infected the computer:
  1. Add the necessary permissions to the SQLite database
  2. Open a hidden (Windows), minimized (Linux) browser window
  3. Navigate the malicious website
  4. Extract the sensitive information (geolocation, audio, video, etc.)

→ Missing permission protection (against data alteration)
Demo #02
Defenses & Future works
Limitations

Proof-of-concept status

*Limited compatibility (macOS and Android)*

*Bookmarks bar and window resize can break attack*

*Increase the excitement of the game*

• Investigate permissions on other systems (iOS, Android, etc.)
Summary

Threats & Attacks  Technical Details  Experiments

![Image of iPad Mini]

HTML5 Permission API

// Check for Geolocation API permissions
navigator.permissions.query(name: 'geolocation')
.then(function(permissionStatus)
console.log('geolocation permission state is', permissionStatus.state);
permissionStatus.onchange = function()
console.log('geolocation permission state has changed to ', this.state);
});

Controlled Experiment

<table>
<thead>
<tr>
<th>Familiarity</th>
</tr>
</thead>
</table>
| % of people using dialog
| % of people trusted |

![Graph showing results of controlled experiment]