Pride and Prejudice in **Progressive Web Apps:**
Abusing Native App-like Features in Web Applications

Jiyeon Lee, Hayeon Kim, Junghwan Park,
Insik Shin, Sooel Son

School of Computing,
Graduate School of Information Security

KAIST
Limitations of Web Apps

• Users spend most of time in native apps
• Reasons:
  - Heavily depend on network connection
  - Low user engagement

Source: comScore Mobile Metrix, U.S., Age 18+, June 2016
Limitations of Native Apps

• App usage is highly concentrated

• Reasons:
  - High cost
  - Difficult to share

Source: comScore Mobile Metrix, U.S., Age 18+, June 2016
Progressive Web Apps (PWAs)

• Introduced by Google in 2015
• Three design goals: reliable, fast, engaging
• Success stories
  - Twitter Lite
  - Financial Times
  - Forbes

Add to Home Screen  Offline Browsing  Push Notifications
Progressive Web Apps (PWAs)

- Introduced by Google in 2015

**Core Components:**

1) *Service Worker*
2) *Cache*
3) *Push*
This Study

• We addressed the security and privacy risks to PWAs

Vulnerabilities:

1) Service Worker → Cryptocurrency Mining
2) Cache → Inferring User’s Browsing History
3) Push → Phishing Attack
Technology behind PWAs: Service Worker

• HTML5 Web standard technology
• Supported by most browsers:
  - Firefox 44+, Chrome 45+, Edge 17+, Opera 32+
• Only usable on HTTPS websites
• Able to run in the background even when a user leaves a website
Offline Browsing

- **Cache** is an origin-bounded local storage
- Accessible regardless of the network status
- Provides programmable offline interfaces with Service Worker
Web Push Notifications

• Re-engaging users with customized content
• Can be received by Service Worker **even if the browser is closed**
How Many PWAs Exist in the Wild?

• A PWA is a website that registers Service Worker
• Collected from the Alexa top 100,000 websites

<table>
<thead>
<tr>
<th>Features Used</th>
<th>Number of websites</th>
</tr>
</thead>
<tbody>
<tr>
<td>Push</td>
<td>3,351 (80.5%)</td>
</tr>
<tr>
<td>Cache</td>
<td>513 (12.3%)</td>
</tr>
<tr>
<td>Both</td>
<td>196 (4.7%)</td>
</tr>
<tr>
<td>Others</td>
<td>495 (11.9%)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>4,163 (100%)</strong></td>
</tr>
</tbody>
</table>
I-1. Phishing Risks of Web Push
General Appearance of Web Push
General Appearance of Web Push

- New article on mobiForge: Why HTML5 is killing Flash: it’s the devices, people!
- Credit Card: Did you make a $1,000,000 purchase at Dr. Evil Inc with VISA *1234?
- Running out of dog food?
- Don’t let the weekend pass by!
- NaN new comment(s)
- Warehouse Sale Starts Now!
- Mike, get those shoes today

INTERACTION COLUMNS:
- ICON
- TITLE
- BODY
- DOMAIN
Sender Can Customize,

- New article on mobiForge
  Why HTML5 is killing Flash: it’s the devices, people!
  mobiforge.com

- Running out of dog food?
  It’s been a month since your last purchase. Make sure your dog has something to eat!
  yourzoooshop.com

- Don't let the weekend pass by!
  Try suggesting one of your activities like ...
  notify.tech

- NaN new comment(s)
  Hey! There are NaN new comments in the Harsh Agrawal
  www.smartmoderation.com

- Warehouse Sale Starts Now!
  Save up to 20% off all products with promo code "SAVE". Click for details
  aimtel.com

- Mike, get those shoes today
  You are eligible for a 5% discount for the next 24hrs
  dayforthbeach.webpu.sh

- Credit Card
  Did you make a $1,000,000 purchase at Dr. Evil Inc with VISA *1234?
  localhost:8000
Sender Can Not Customize,

- A domain name is the only element representing the source of a push message
Vulnerabilities We Found

- The environments that **do not display domains**
  - Firefox on GNOME, Ubuntu MATE, Cinnamon, Budgie, and Pantheon
  - Samsung Internet, Firefox on Android
- Causes phishing risks

<table>
<thead>
<tr>
<th>Push without domain</th>
<th>Push with domain</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Firefox</strong></td>
<td><strong>Chrome</strong></td>
</tr>
<tr>
<td>Gmail Account Manager</td>
<td>Gmail Account Manager</td>
</tr>
<tr>
<td>11:12 AM</td>
<td>11:06 AM</td>
</tr>
<tr>
<td>Your Gmail account needs to</td>
<td>test.kaist.ac.kr</td>
</tr>
<tr>
<td>be validated.</td>
<td></td>
</tr>
<tr>
<td><strong>Samsung Internet</strong></td>
<td></td>
</tr>
<tr>
<td>Gmail Account Manager</td>
<td></td>
</tr>
<tr>
<td>11:10 AM</td>
<td></td>
</tr>
<tr>
<td>Your Gmail account needs to</td>
<td></td>
</tr>
<tr>
<td>be validated.</td>
<td></td>
</tr>
</tbody>
</table>
I-II. Phishing risks of Third-Party Push Libraries
Emerging Third-party Push Services

• Enable website owners to use push features
• Provide useful features:
  - Scheduling push notifications, Reporting the statistics of subscribers, 
    **Supporting HTTP websites** 

Image Source: https://sendpulse.com/features/webpush
How push is Supported on HTTP Sites
How push is Supported on HTTP Sites
How push is Supported on HTTP Sites

A css-styled permission dialog that is drawn by the library

An actual permission dialog that a browser asks
How push is Supported on HTTP Sites

An address of website that user visits

A HTTPS domain that library creates
Permission Delegation Attack

• A network attacker can redirect users to an attacker-controlled website
• A visitor has no clue why she is redirected to a different domain
I-III. Domain Name Spoofing Attack of Web Push Notifications
Web Push in Detail

1. Asks Permission
   - Yes

2. Subscribe to push service

3. Generated endpoint URL returned

4. The endpoint sent to the web server

5. The endpoint stored

6. Push message sent to the endpoint URL

7. Push message sent to the browser

8. Push message sent to service worker
Web Push in Detail

1. Asks Permission
   - Yes

2. Subscribe to push service

3. Generated endpoint URL returned

4. The endpoint sent to the web server

5. The endpoint stored

6. Push message sent to the endpoint URL

7. Push message sent to the browser

8. Push message sent to service worker

An example of endpointURL:
https://fcm.googleapis.com/fcm/send/dTb6ILBpUYs:A
PA91bGX_Xa91bizHC-
ol0qF9fj7f2u9lt3mExBdbhGsE0zCuXkPJioWDgo4wf1m
TfZYgqX_-sVWRabWqx3GB9XiA9hsUf-
gVnwkkbD8oDLAUlhScYYrmeSZaricyZv3gq3hbzjh48Ad
Web Push in Detail

1. Asks Permission  
   - Yes

2. Subscribe to push service

3. Generated endpoint URL returned

4. The endpoint sent to the web server

5. The endpoint stored

6. Push message sent to the endpoint URL

7. Push message sent to the browser

8. Push message sent to service worker
Web Push in Detail

1. Asks Permission
   - Yes

2. Subscribe to push service

3. Generated endpoint URL returned

4. The endpoint sent to the web server

5. The endpoint stored

6. Push message sent to the endpoint URL

7. Push message sent to the browser

8. Push message sent to service worker
Web Push in Detail

1. Asks Permission
   - Yes

2. Subscribe to
   - The EndpointURL URL returned

3. Generated endpoint URL returned

4. The endpoint sent to the web server

5. The endpoint stored

6. Push message sent to the endpoint

7. Push message sent to the browser

8. Push message sent to service worker

The EndpointURL is confidential information!
Web Push Protocol: VAPID

1. Permission asking
   - Yes

2. Subscribe to push service

3. Get generated endpoint and encryption key

4. Send the endpoint and encryption key to the web server

5. Store the endpoint and encryption keys

6. Send the encrypted payload to the endpoint

7. Payload received on the URL is sent to the browser

8. The payload is decrypted and sent to the service worker

- Designed to authenticate web servers
- Utilizes asymmetrical key pairs
  - Without a private key, cannot send push messages

Public Key

Private Key

• Designed to authenticate web servers
• Utilizes asymmetrical key pairs
  - Without a private key, cannot send push messages
VAPID in the Wild

1. Permission asking
2. Subscribe to push service
3. Get generated endpoint and encryption key
4. Send the endpoint and encryption key to the web server
5. Store the endpoint and encryption keys
6. Send the encrypted payload to the endpoint
7. Payload received on the URL is sent to the browser
8. The payload is decrypted and sent to the service worker

<table>
<thead>
<tr>
<th>Third-party Library</th>
<th>VAPID</th>
</tr>
</thead>
<tbody>
<tr>
<td>SnedPulse</td>
<td>X</td>
</tr>
<tr>
<td>Izooto</td>
<td>X</td>
</tr>
<tr>
<td>Pushwoosh</td>
<td>X</td>
</tr>
<tr>
<td>Foxpush</td>
<td>X</td>
</tr>
<tr>
<td>OneSignal</td>
<td>✔</td>
</tr>
<tr>
<td>Pushcrew</td>
<td>X</td>
</tr>
<tr>
<td>Pushengage</td>
<td>X</td>
</tr>
<tr>
<td>Urbanairship</td>
<td>✔</td>
</tr>
</tbody>
</table>
Domain Spoofing Attack

1. Permission asking
   - Yes

2. Subscribe to push service

3. Get generated endpoint and encryption key

4. The endpoint sent to the web server over HTTP

5. Store the endpoint and encryption keys

6. Send the encrypted payload to the endpoint

7. Payload received and sent to the browser

8. The payload is decrypted and sent to the service worker

A push with spoofed domain, “kirannewsagency.iz.do”
Why **Phishing via Web Push** Matters?

- Difficult to determine the origin of messages
- An attacker can send push messages at any time

Real-world phishing
II. User Privacy Leak via Offline Usage
History Sniffing Attack

• Critical privacy threat
  - E. Felten at al., Timing Attacks on Web Privacy [CCS 2000]
  - S. Son at al., What Mobile Ads Know About Mobile Users [NDSS 2016]

• Can leak personal information
History Sniffing Attack on PWAs

• A new side channel attack that exploits Cache
History Sniffing Attack on PWAs

• A new side channel attack that exploits **Cache**
• How it works:
History Sniffing Attack on PWAs

• A new side channel attack that exploits **Cache**

• How it works:

  1. A victim opens the attacking PWA offline
History Sniffing Attack on PWAs

• A new side channel attack that exploits *Cache*

• How it works:
  1. A victim opens the attacking PWA offline
  2. An *onload* event will only be triggered if victims have visited target PWAs
History Sniffing Attack on PWAs

• A new side channel attack that exploits **Cache**
• How it works:
  1. A victim opens the attacking PWA offline
  2. An *onload* event will only be triggered if victims have visited target PWAs

**Advantages:**
1) *Accuracy*
2) *No outgoing requests*
Consequences of History Sniffing Attack

• Vulnerable Browser: Firefox 59.0.2
• X-Frame-Options, CSP, and Frame Busting are effective to defense

<table>
<thead>
<tr>
<th>Offline Cache Attack</th>
<th># of Websites</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vulnerable</td>
<td>187 (36.5%)</td>
</tr>
<tr>
<td>X-Frame-Options</td>
<td>132 (25.7%)</td>
</tr>
<tr>
<td>CSP</td>
<td>22 (4.3%)</td>
</tr>
<tr>
<td>Frame Busting</td>
<td>10 (1.9%)</td>
</tr>
<tr>
<td>Others</td>
<td>162 (31.6%)</td>
</tr>
<tr>
<td>Total</td>
<td>513 (100%)</td>
</tr>
</tbody>
</table>

• Safari manages cache separately from the first-party
III. Cryptocurrency Mining Attack
Using Service Worker
Cryptocurrency Mining in the Web

• *CoinHive* is a popular JavaScript cryptocurrency mining service
• Main Limitation:
  - Stops when user leaves
Cryptocurrency Mining Attack

• *CoinHive* is a popular JavaScript cryptocurrency mining service
• Main Limitation:
  - Stops when user leaves

• Introducing cryptocurrency mining attack *using Service Worker*

**Advantages:**
1) *Stealthy*
2) *Lasting Longer*
Cryptocurrency Mining Attack

• Technical challenges:
  - Service Worker becomes idle
  - Service Worker cannot use WebSocket
Cryptocurrency Mining Attack

• Technical challenges:
  - Service Worker becomes idle
  - Service Worker cannot use WebSocket

• Solution:
  - *Push notifications*
Cryptocurrency Mining Attack

• Two tricks:
  - Non-visible push
  - Re-subscription

• Different browsers have different policies:

<table>
<thead>
<tr>
<th></th>
<th>Whale</th>
<th>Brave</th>
<th>Samsung Internet</th>
<th>Opera</th>
<th>Chrome</th>
<th>Edge</th>
<th>Firefox</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-visible push</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Re-subscription in the background</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>

Most stealthy!
Cryptocurrency Mining Results

- Mined Monero coins for 24 hours using a single service worker

<table>
<thead>
<tr>
<th>Browser</th>
<th>Environment</th>
<th>Number of Solved Hashes (24h)</th>
<th>Amount of Monero (24h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chrome 65</td>
<td>Window 10 Desktop (3.6GHz Intel Core i7, 16GB)</td>
<td>225,024</td>
<td>0.00001266</td>
</tr>
<tr>
<td>Firefox 69</td>
<td>Window 10 Desktop (3.6GHz Intel Core i7, 16GB)</td>
<td>195,840</td>
<td>0.00001119</td>
</tr>
<tr>
<td>Chrome 65</td>
<td>Android 8.0 Google Pixel</td>
<td>50,176</td>
<td>0.00000282</td>
</tr>
<tr>
<td>Chrome 65</td>
<td>macOS High Sierra 10.13.4 (1.3GHz Intel Core i5, 8GB)</td>
<td>138,496</td>
<td>0.00000778</td>
</tr>
</tbody>
</table>

- The more victims, the more lucrative this attack is
Lessons Learned

• Web Push requires careful use
  - adopt VAPID
  - treat EndpointURL as confidential information
• Well known defenses are helpful
• Better design for supporting web push for HTTP websites is Required
Conclusion

• The first in-depth study of PWAs
• Proposed novel attacks that abuse fundamental features of PWAs
• Provided mitigating recommendations
• Reported findings to corresponding vendors
• All demonstrations can be found at https://github.com/spostman/ppp-ccs2018
Thank You!

Q&A
Consequences of Permission Delegation Attack

<table>
<thead>
<tr>
<th>Third-party Library</th>
<th>Attack Success</th>
<th>Number of affected HTTP sites</th>
</tr>
</thead>
<tbody>
<tr>
<td>SnedPulse</td>
<td>✔️</td>
<td>93</td>
</tr>
<tr>
<td>Izooto</td>
<td>✔️</td>
<td>18</td>
</tr>
<tr>
<td>Pushwoosh</td>
<td>✔️</td>
<td>4</td>
</tr>
<tr>
<td>Foxpush</td>
<td>✔️</td>
<td>1</td>
</tr>
<tr>
<td>OneSignal</td>
<td>✗</td>
<td>528</td>
</tr>
<tr>
<td>Pushcrew</td>
<td>✗</td>
<td>31</td>
</tr>
<tr>
<td>Pushengage</td>
<td>✗</td>
<td>19</td>
</tr>
<tr>
<td>Urbanairship</td>
<td>✗</td>
<td>2</td>
</tr>
</tbody>
</table>

A permission delegation attack against http://www.koimoi.com
## Domain Spoofing Attack Implication

<table>
<thead>
<tr>
<th>Third-party Library</th>
<th>Attack Success</th>
<th>Number of affected HTTP sites</th>
</tr>
</thead>
<tbody>
<tr>
<td>SnedPulse</td>
<td>✓</td>
<td>93</td>
</tr>
<tr>
<td>Izooto</td>
<td>✓</td>
<td>18</td>
</tr>
<tr>
<td>Pushwoosh</td>
<td>✓</td>
<td>4</td>
</tr>
<tr>
<td>Foxpush</td>
<td>✗</td>
<td>1</td>
</tr>
<tr>
<td>OneSignal</td>
<td>✗</td>
<td>528</td>
</tr>
<tr>
<td>Pushcrew</td>
<td>✗</td>
<td>31</td>
</tr>
<tr>
<td>Pushengage</td>
<td>✗</td>
<td>19</td>
</tr>
<tr>
<td>Urbanairship</td>
<td>✗</td>
<td>2</td>
</tr>
</tbody>
</table>

A push with spoofed domain, “kirannewsagency.iz.do”

A push with spoofed domain, “afn.sendpulse.com”
Web Push Protocol: VAPID

1. Permission asking
Yes

2. Subscribe to push service

3. Get generated endpoint and encryption key

4. Send the endpoint and encryption key to the web server

5. Store the endpoint and encryption keys

6. Send the encrypted payload to the endpoint

7. Payload received on the URL is sent to the browser

8. The payload is decrypted and sent to the service worker
Web Push Protocol: VAPID

1. Permission asking
   - Yes

2. Subscribe to push service

3. Get generated endpoint and encryption key

4. Send the endpoint and encryption key to the web server

5. Store the endpoint and encryption keys

6. Send the encrypted payload to the endpoint

7. Payload received on the URL is sent to the browser

8. The payload is decrypted and sent to the service worker

---

Public Key

WEB SERVER

Private Key

PUSH SERVER

SERVICE WORKER
Domain Spoofing Attack

1. Permission asking
   - Yes
2. Subscribe to push service
3. Get generated endpoint and encryption key
4. Send the endpoint and encryption key to the web server
5. Store the endpoint and encryption keys
6. Send the encrypted payload to the endpoint
7. Payload received on the URL is sent to the browser
8. The payload is decrypted and sent to the service worker

GET script.js

Script.js is sent

THIRD-PARTY SERVER

SERVICE WORKER
Domain Spoofing Attack

1. Permission asking
2. Subscribe to push service
3. Get generated endpoint and encryption key
4. Send the endpoint and encryption key to the web server
5. Store the endpoint and encryption keys
6. Send the encrypted payload to the endpoint
7. Payload received on the URL is sent to the browser
8. The payload is decrypted and sent to the service worker

A push with spoofed domain, “afn.sendpulse.com”

GET script.js

Script.js is sent over HTTP

var n="https://pushdata.sendpulse.com ";

var n="https://attacker.server.com ";