2FA-PP: 2nd Factor Phishing Prevention

15.05.2019
Miami, USA

Enis Ulqinaku, Daniele Lain, Srdjan Čapkun

ETH Zürich
User Authentication Challenges

The phishing email that hacked the account of John Podesta

March 2016

This appears to be the phishing email that hacked Clinton campaign chairman John Podesta's Gmail account. Further, The Clinton campaign's own computer help desk thought it was real email sent by Google, even though the email address had a suspicious "googlemail.com" extension.

- The John Podesta emails released by WikiLeaks
Two Factor Authentication (2FA)
Two Factor Authentication (2FA)

1. user opens a website
Two Factor Authentication (2FA)

1. user opens a website

1. client connected
Two Factor Authentication (2FA)

End User

1. user opens a website
2. enters credentials

Server

1. client connected
2. verifies credentials
Two Factor Authentication (2FA)

End User

1. user opens a website
2. enters credentials

Server

1. client connected
2. verifies credentials

what is the OTP?

credentials
Two Factor Authentication (2FA)

End User

1. user opens a website
2. enters credentials

Server

1. client connected
2. verifies credentials

what is the OTP?

OTP
Two Factor Authentication (2FA)

End User
1. user opens a website
2. enters credentials

Server
1. client connected
2. verifies credentials
3. verifies OTP

OTP
Two Factor Authentication (2FA)

End User
1. user opens a website
2. enters credentials
3. gets logged in

Server
1. client connected
2. verifies credentials
3. verifies OTP
Runtime Phishing Attacks

End User

Adversary

Gmail Server
Runtime Phishing Attacks

1. gets phished

End User

attack.com

Adversary

Gmail Server
Runtime Phishing Attacks

1. gets phished
2. enters credentials
Runtime Phishing Attacks

1. gets phished
2. enters credentials

1. attacker opens real-gmail

1. client connected
Runtime Phishing Attacks

1. End User gets phished
2. End User enters credentials

1. Adversary attacker opens real-gmail
2. Adversary forwards credentials

1. Gmail Server client connected
Runtime Phishing Attacks

1. \textbf{End User} gets phished
2. \textbf{End User} enters credentials

1. \textbf{Adversary} attacker opens real-gmail
2. \textbf{Adversary} forwards credentials

1. \textbf{Gmail Server} client connected
2. \textbf{Gmail Server} verifies credentials

\textit{what is OTP?}
Runtime Phishing Attacks

1. End User
   - gets phished
   - enters credentials

2. Adversary
   - attacker opens real-gmail
   - forwards credentials
   - forwards the question
   - what is OTP?

3. Gmail Server
   - client connected
   - verifies credentials
   - credentials
   - what is OTP?
Runtime Phishing Attacks

1. End User gets phished
2. End User enters credentials

1. Adversary attacker opens real-gmail
2. Adversary forwards credentials
3. Adversary forwards the question

1. Gmail Server client connected
2. Gmail Server verifies credentials

what is OTP?

attack.com
Runtime Phishing Attacks

1. End User gets phished
2. Enters credentials

1. Adversary attacker opens real-gmail
2. Forwards credentials
3. Forwards the question
4. Forwards the OTP

1. Gmail Server client connected
2. Verifies credentials

what is OTP?
Runtime Phishing Attacks

End User

1. gets phished
2. enters credentials

Adversary

1. attacker opens real-gmail
2. forwards credentials
3. forwards the question
4. forwards the OTP

Gmail Server

1. client connected
2. verifies credentials
3. verifies OTP

what is OTP?

attack.com
Runtime Phishing Attacks

1. gets phished
2. enters credentials

End User

1. attacker opens real-gmail
2. forwards credentials
3. forwards the question
4. forwards the OTP
5. gets logged in

Adversary

1. client connected
2. verifies credentials
3. verifies OTP

Gmail Server
Runtime Phishing Attacks

1. End User gets phished
2. End User enters credentials
3. Adversary attacker opens real-gmail
4. Adversary forwards credentials
5. Adversary forwards the question
6. Adversary forwards the OTP
7. Adversary gets logged in
8. Gmail Server client connected
9. Gmail Server verifies credentials
10. Gmail Server verifies OTP

what is OTP?

https://github.com/kgretzky/evilginx2
Our Goal

End User

Adversary

Server

attack.com
Our Goal

End User

Adversary

Server

attack.com
Our Goal

End User

attack.com

challenge

response

Adversary

Server
Our Goal

End User

Adversary

Server

challenge

response

attack.com

challenge

response
Our Goal

End User

Adversary

Server

Targets:
- High Usability
- Deployability
- Security (Machine Confirmation)
The Idea

End User

Server
The Idea

1. Encrypt JS code
2. Deliver the cipher text
The Idea

1. Encrypt JS code
2. Deliver the cipher text
The Idea

1. Encrypt JS code
2. Deliver the cipher text
The Idea

1. Encrypt JS code
2. Deliver the cipher text
The Idea

1. Encrypt JS code
2. Deliver the cipher text
The Idea

End User

Server

1. Encrypt JS code
2. Deliver the cipher text

key

response

Proximity
Correct Domain
Protocol Overview

1. enter credentials

2. submit credentials

Verify
Protocol Overview

1. enter credentials
2. submit credentials
3. deliver the encrypted code

Verify
Protocol Overview

1. Enter credentials
2. Submit credentials
3. Deliver the encrypted code
4. Request the decryption key

Verify
Protocol Overview

1. enter credentials
2. submit credentials
3. deliver the encrypted code
4. request the decryption key
5. send the key, start the clock
6. send response

Verify

Execute JS code
Protocol Overview

1. Enter credentials
2. Submit credentials
3. Deliver the encrypted code
4. Request the decryption key
5. Send the key, start the clock
6. Send response
7. Verify

Execute JS code
Protocol Overview

1. Enter credentials
2. Submit credentials
3. Deliver the encrypted code
4. Request the decryption key
5. Send the key, start the clock
6. Send response
7. Verify
8. Accept/reject login
8a. Alert

Execute JS code
Login accepted set cookie
JavaScript Code
Verify the correct domain.
Verify the correct domain.

Unique code for every attempt
JavaScript Code

- Verify the correct domain.
- Unique code for every attempt
- Valid only for few milliseconds
JavaScript Code

- Verify the correct domain.
- Unique code for every attempt
- Valid only for few milliseconds
- Should resist tamper from automated tools in JS (obfuscated)
  - Limited Attacker capabilities
JavaScript Code

- Verify the correct domain.
- Unique code for every attempt
- Valid only for few milliseconds
- Should resist tamper from automated tools in JS (obfuscated)
  - Limited Attacker capabilities
- Hard for the attacker to extract the nonce
Verify the correct domain.

Unique code for every attempt

Valid only for few milliseconds

Should resist tamper from automated tools in JS (obfuscated)

- Limited Attacker capabilities

Hard for the attacker to extract the nonce

Execution flow depends on runtime values
Obfuscated Code: One Block Example

```javascript
v1 = v2 = window.location;
v3 = {href: "a-long-random-string"};

function f0(x) {
    partial = partial_1 = partial_2 = ""; // BLOCK START
    if (824768 < hash(x) < 1893859) {
        v11 = v2; v2 = v3; v3 = v11; // Mixing Phase
        v5 = "hello";
        if (v11.v5 == v5) {
            // check whether v11 is a the window object
            // Some of the approaches to make the code context-free
            if (partial_2 += v3['href'][2] && v2['x']) {
                for(((partial_1 += v2.href[1]), partial_2 += v1.href[5]), partial_1 += v11['href'][2], 0);
                partial = partial_1 + partial_2;
                // Continue execution
                f6(hash(x) ⊕ partial);
            } else { // ... Similar to the above block }
        } else { // ... Similar to the above block }
    } ...
```
Relay Attack

End User

Adversary

Server

attack.com
Relay Attack

End User

[Image of a laptop with a web browser showing a website]

Adversary

[Image of a hacker in a hoodie]

Server

[Image of a server with a secure connection]

attack.com
Relay Attack

End User

Adversary

Server

attack.com
Relay Attack

End User

Adversary

Server

attack.com

key
Relay Attack

End User

Adversary

Server

attack.com

key

key
Relay Attack

End User → Adversary → Server

key

attack.com

delay
Relay Attack

End User → Adversary

key → reply → key

attack.com

Adversary → Server

Server
Relay Attack

End User → Adversary → Server

Key
Reply

End User

Key
Reply

Adversary

Key
Reply

Server

attack.com
Experiment Setup

- A malicious website
- Two configurations: 1-one hop, 2-hops forwarding
- Evaluate the attacker’s success rate from different locations
- The adversary owns a fast Gigabit Ethernet
- No VPN/Proxy employed by the attacker
Results

Victim location regarding to the attacker:

1. different countries.
2. the same city.
3. the same public network.
4. in the attacker’s hotspot.
Machine Verification

End User

Server
Machine Verification

End User

Server
Machine Verification

End User

Server
Machine Verification

End User

key

Server
Machine Verification

End User

Server

key
Machine Verification

End User

Server

key

reply
Machine Verification

End User

key

reply

Server

MAC
60-E3-5C-FE-41-D3
Machine Verification

End User

Attack.com

Submits to Adversary

Adversary

Forward to Gmail Server

Gmail Server
Machine Verification

End User

Adversary

Gmail Server

key

reply

MAC
60-E3-5C-FE-41-D3

submit

forward

attack.com
Conclusions

- Usability: fast, effective, no significant burden to the user.
- Deployability: integrate well with existing infrastructure.
- Security: produce valid 2FA credentials only for legitimate websites, confirm if user authenticated previously in the same PC.
Thank you!

enis.ulqinaku@inf.ethz.ch

Danke
Faleminderit
Grazie
Hvala
Merci