You shall not password!

An extensive analysis of multi-factor authentication protocols

Charlie Jacomme, Steve Kremer
CSF 2018
Introduction
Secure Authentication:
Every accepted login by the server and coming from some computer has been initiated on the very same computer by the user.

Common solution: login / password
Passwords are bad

Passwords are compromised:

- Database leaks
- Phishing
- Keyloggers
Passwords are bad

Passwords are compromised:

- Database leaks
- Phishing
- Keyloggers

Everybody uses the same password everywhere!
Passwords are bad

Passwords are compromised:

- Database leaks
- Phishing
- Keyloggers

Everybody uses the same weak password everywhere!
Passwords are bad

Passwords are compromised:

- Database leaks
- Phishing
- Keyloggers

Everybody uses the same weak password everywhere!
"1234", "password", "qwerty"
Passwords are bad

Passwords are compromised:

- Database leaks
- Phishing
- Keyloggers

Everybody uses the same weak password everywhere!
"1234", "password", "qwerty"

Requirement to add special characters or on length does not work
Passwords are bad

Passwords are compromised:

- Database leaks
- Phishing
- Keyloggers

Everybody uses the same weak password everywhere!

"1234", "password", "qwerty"

Requirement to add special characters or on length does not work

"123456!", "p@ssword1", "Qwerty"
The current solution

Use a second factor to confirm login, either a smartphone or a dedicated token.
The current solution
Use a second factor to confirm login, either a smartphone or a dedicated token.

Protocols we studied:

- Google 2 Step (Verification code, One Tap, Double Tap)
- FIDO’s U2F (Google, Facebook, Github, Dropbox, ...)

Main ideas

A case study of Google 2 Step and FIDO’s U2F

- Many different threat models (malwares, phishing, human errors...)
- Automated analysis of all scenarios

→ 6 172 (non-redundant) scenarios analysed by PROVERIF in 8 minutes
Presentation of the protocols
Google 2 Step - Verification Code

id, pass → TLS communication

id, pass → id, pass

id, pass → id, pass

TLS communication → id, pass

TLS communication → id, pass

TLS communication → id, pass

TLS communication → id, pass
Google 2 Step - Verification Code

id, pass

→ TLS communication
Google 2 Step - Verification Code

TLS communication
Google 2 Step - One Tap
Google 2 Step - One Tap

id, pass

id, pass

id, pass

id, pass

yes

fpr, ok?

fpr, ok?

fpr, ok?

fpr, ok?

token, fpr

token, fpr

token, fpr

fpr : IP, location, OS,...
A token with cryptographic capabilities

- A public key is registered server side.
- On login, a challenge containing a random nonce, the origin and the TLS sid is signed.
An option provided by major companies (Google, Facebook,...):

I trust this computer = disable second factor
An option provided by major companies (Google, Facebook,...):

I trust this computer = disable second factor

It must be taken into account in the analysis
Threat model
First hypothesis

The user password has been compromised
First hypothesis
The user password has been compromised

Goal
Consider many different scenarios:

- Malware on the computer
- Malware on the phone
- Human errors (Phishing, No Compare)
- Fingerprint Spoofing

What guarantees from different protocols under different threats?
Device = set of interfaces

Modelling Malwares
Modelling Malwares

Device = set of interfaces

Access levels
Read Only or Read Write

Out RW
✓

In RW	Out RO
✓

In RO
✓

NA
Scenarios

Notations

- Malware: $\mathcal{M}_{\text{interf}}^{\text{in:acc1,out:acc2}}$
- Phising: PH
- Fingerprint Spoofing: FS
- No Compare: NC
Scenarios

Notations

- Malware: $M_{\text{interf}}^{\text{in:acc1,out:acc2}}$
- Phising: PH
- Fingerprint Spoofing: FS
- No Compare: NC

Examples

- Keylogger: $M_{\text{usb}}^{\text{in:RO}}$
- Wifi Hotspot: FS PH
- Broken TLS encryption: $M_{\text{tls}}^{\text{io:RW}}$
Modeling in Proverif
TLS modeling

- A set of identities: $id_{server}, id_{user's computer}, \ldots$
- A private function symbol $tls$
TLS modeling

- A set of identities: \( id_{\text{server}} \), \( id_{\text{user's computer}} \), ...
- A private function symbol \( tls \)

\[
\text{TLS} := \\
\text{Asynchronous communications over channel } tls(id_{\text{client}}, id_{\text{server}})
\]
TLS modeling

- A set of identities: \( id_{server}, id_{user's computer}, \ldots \)
- A private function symbol \( tls \)

\[
\text{TLS} := \text{Asynchronous communications over channel } tls(id_{client}, id_{server})
\]

If \( id_{client} \) or \( id_{server} \) is compromised, we give \( tls(id_{client}, id_{server}) \) to the attacker
Read only access to some channel \( ch \):

\[
\text{in}(ch, x).P \rightarrow \text{in}(ch, x).\text{out}(a, x).P
\]

or

\[
\text{out}(ch, x).P \rightarrow \text{out}(a, x).\text{out}(ch, x).P
\]
Malwares

Read only access to some channel $ch$:

$$\text{in}(ch, x).P \rightarrow \text{in}(ch, x).\text{out}(a, x).P$$

or

$$\text{out}(ch, x).P \rightarrow \text{out}(a, x).\text{out}(ch, x).P$$

Read write access to $ch$:

$$P \rightarrow \text{out}(a, ch).P$$
Human errors

No compare
Remove some checks

Phishing
The server’s url ($id_{server}$) is chosen by the attacker.
→ The human may check or not that it is indeed the server he wishes to contact.
Fingerprint Spoofing

Fingerprint
A function symbol $fpr(id)$

→ a server may obtain $fpr(id_{client})$ from $tls(id_{client}, id_{server})$
Fingerprint Spooing

Fingerprint
A function symbol $fpr(id)$
→ a server may obtain $fpr(id_{client})$ from $tls(id_{client}, id_{server})$

Spoofing

$$fpr(spoof_{fpr}(fpr(c))) = fpr(c)$$
Analysis
Three types of login

- **untrusted login**: login on an untrusted computer
- **trusted login**: login on a trusted computer; sets “trust this computer” option
- **cookie login**: login after “trust this computer” option enabled
Properties

Three types of login

- **untrusted login** login on an untrusted computer
- **trusted login** login on a trusted computer; sets “trust this computer” option
- **cookie login** login after “trust this computer” option enabled

Three properties

\[ \text{accept}_x(id) \implies_{\text{inj}} \text{request}_x(id) \quad x \in \{u, t, c\} \]

*Every accepted login was preceded by a distinct login request by the human.*
One file = one protocol with all scenarios

```plaintext
let Device =
    in(d_in,(token));
    #if defined(D_I_RO) && !defined(D_I_RW)
      out(a,(token));
      #endif
    out(d_out,(token))
```

Methodology

A bash script

- takes a combination of attacker capabilities as input
- generates the proverif file

A python script

- runs proverif for all pertinent combinations of scenarios
- generate the result table
## Analysis of Google 2 step protocols

<table>
<thead>
<tr>
<th>Threat Scenarios</th>
<th>g2V</th>
<th>g2OT</th>
<th>g2OT&lt;sup&gt;fpr&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>PH</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>NC</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>FS</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>PH, NC</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>PH, FS</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>

- $M_{\text{dev in: RO}}$
- $M_{\text{t-dis io: RO}}$
- $M_{\text{t-tls io: RO}}$
- $M_{\text{t-usb in: RO}}$
- $M_{\text{dev in: RW}}$
- $M_{\text{t-tls io: RW}}$
- $M_{\text{t-usb in: RW}}$

...
• It is secure if the attacker only knows the password
• in any other cases...
Attack under Keylogger or Phishing or Malware
• Without fingerprint, never secure: one can easily validate an attacker session
Adding the display

**Recommendation:**
Display (via SMS or on the smartphone screen) additional info:

- fingerprint (IP, locations, computer model).
- the type of login desired.

**Benefits:**

- avoids attacks changing the login type (e.g. replacing an untrusted, by trusted login)
- avoids attacks where attacker is able to spoof a fingerprint
## Adding the display

<table>
<thead>
<tr>
<th>Threat Scenarios</th>
<th>g2V&lt;sup&gt;fpr&lt;/sup&gt;</th>
<th>g2V&lt;sup&gt;dis&lt;/sup&gt;</th>
<th>g2OT&lt;sup&gt;dis&lt;/sup&gt;</th>
<th>g2DT&lt;sup&gt;dis&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>PH</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>PH FS</td>
<td>×</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>PH FS M&lt;sub&gt;tls&lt;/sub&gt; t&lt;sub&gt;tls&lt;/sub&gt;</td>
<td>×</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>PH FS M&lt;sub&gt;tls&lt;/sub&gt; t&lt;sub&gt;usb&lt;/sub&gt; in&lt;sub&gt;RO&lt;/sub&gt;</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>PH FS M&lt;sub&gt;tls&lt;/sub&gt; t&lt;sub&gt;dis&lt;/sub&gt; io&lt;sub&gt;RW&lt;/sub&gt;</td>
<td>×</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>M&lt;sub&gt;tls&lt;/sub&gt; t&lt;sub&gt;tls&lt;/sub&gt; io&lt;sub&gt;RO&lt;/sub&gt;</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>M&lt;sub&gt;tls&lt;/sub&gt; t&lt;sub&gt;usb&lt;/sub&gt; io&lt;sub&gt;RO&lt;/sub&gt;</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>M&lt;sub&gt;tls&lt;/sub&gt; t&lt;sub&gt;usb&lt;/sub&gt; in&lt;sub&gt;RO&lt;/sub&gt;</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>M&lt;sub&gt;tls&lt;/sub&gt; t&lt;sub&gt;dis&lt;/sub&gt; in&lt;sub&gt;RW&lt;/sub&gt;</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>M&lt;sub&gt;tls&lt;/sub&gt; t&lt;sub&gt;tls&lt;/sub&gt; in&lt;sub&gt;RW&lt;/sub&gt;</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FS M&lt;sub&gt;tls&lt;/sub&gt; t&lt;sub&gt;tls&lt;/sub&gt; in&lt;sub&gt;RO&lt;/sub&gt;</td>
<td>×</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>FS M&lt;sub&gt;tls&lt;/sub&gt; t&lt;sub&gt;usb&lt;/sub&gt; in&lt;sub&gt;RO&lt;/sub&gt;</td>
<td>√</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FS M&lt;sub&gt;tls&lt;/sub&gt; t&lt;sub&gt;dis&lt;/sub&gt; in&lt;sub&gt;RW&lt;/sub&gt;</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FS M&lt;sub&gt;tls&lt;/sub&gt; t&lt;sub&gt;tls&lt;/sub&gt; in&lt;sub&gt;RW&lt;/sub&gt;</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FS M&lt;sub&gt;tls&lt;/sub&gt; t&lt;sub&gt;usb&lt;/sub&gt; in&lt;sub&gt;RW&lt;/sub&gt;</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FS M&lt;sub&gt;tls&lt;/sub&gt; t&lt;sub&gt;dis&lt;/sub&gt; in&lt;sub&gt;RW&lt;/sub&gt;</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FS M&lt;sub&gt;tls&lt;/sub&gt; t&lt;sub&gt;tls&lt;/sub&gt; in&lt;sub&gt;RW&lt;/sub&gt;</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FS M&lt;sub&gt;tls&lt;/sub&gt; t&lt;sub&gt;usb&lt;/sub&gt; in&lt;sub&gt;RW&lt;/sub&gt;</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FS M&lt;sub&gt;tls&lt;/sub&gt; t&lt;sub&gt;dis&lt;/sub&gt; in&lt;sub&gt;RW&lt;/sub&gt;</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FS M&lt;sub&gt;tls&lt;/sub&gt; t&lt;sub&gt;tls&lt;/sub&gt; in&lt;sub&gt;RW&lt;/sub&gt;</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FS M&lt;sub&gt;tls&lt;/sub&gt; t&lt;sub&gt;usb&lt;/sub&gt; in&lt;sub&gt;RW&lt;/sub&gt;</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FS M&lt;sub&gt;tls&lt;/sub&gt; t&lt;sub&gt;dis&lt;/sub&gt; in&lt;sub&gt;RW&lt;/sub&gt;</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FS M&lt;sub&gt;tls&lt;/sub&gt; t&lt;sub&gt;tls&lt;/sub&gt; in&lt;sub&gt;RW&lt;/sub&gt;</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FS M&lt;sub&gt;tls&lt;/sub&gt; t&lt;sub&gt;usb&lt;/sub&gt; in&lt;sub&gt;RW&lt;/sub&gt;</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FS M&lt;sub&gt;tls&lt;/sub&gt; t&lt;sub&gt;dis&lt;/sub&gt; in&lt;sub&gt;RW&lt;/sub&gt;</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FS M&lt;sub&gt;tls&lt;/sub&gt; t&lt;sub&gt;tls&lt;/sub&gt; in&lt;sub&gt;RW&lt;/sub&gt;</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FS M&lt;sub&gt;tls&lt;/sub&gt; t&lt;sub&gt;usb&lt;/sub&gt; in&lt;sub&gt;RW&lt;/sub&gt;</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Pros of U2F

• a possibility of privacy
• strong protection against phishing

Cons of U2F

• no feedback
• not independent from the computer
Conclusions

- Detailed threat model for multi-factor authentication protocols
- Analysis of the full system
- Complete automation using PROVERIF and scripts
- Simple, small modifications (adding info to display) that enhance security