ChromiumOS Printing Update
Printer Working Group F2F May 2022
Agenda

● Review of ChromiumOS
● Projects used in Printing
● Features in Chromium
● Improvements since last year
● New Project: OAuth 2 for IPP
What is ChromiumOS?

- Google’s Open Source operating system for Chromebooks (and other devices)
  - Approximately the same as ChromeOS minus some Google-only parts
- Gentoo derivative
  - Everything is built from source
- Supports a variety of ARM and x86-64 architectures
- Code available at chromium.googlesource.com
Open Source Projects in ChromiumOS

- **CUPS**
  - Print spooling
  - Driverless support
- **cups-filters**
  - gstoraster
  - pdftops
  - foomatic-rip
- **Ghostscript**
- **sane-airscan**: Mopria eSCL scanning
- **SANE**
- **avahi + nss-mdns**: mDNS resolution
- **ippusb_bridge**: local IPP-USB sockets
Features in Chromium

- mDNS detection
- Driverless support
- Matching printers with PPDs
- IPP-USB through local (UNIX domain) sockets
Recent Improvements

● General scalability of existing features
  ○ More PPDs available
  ○ More manufacturer-specific PPD keywords supported
  ○ More automated testing
  ○ Mock printer improvements

● Better sharing of USB devices between printing and scanning

● New feature: OAuth for IPP
OAuth 2 for IPP

1. Scope of the project
2. Security considerations
3. Proposed protocol
4. Project status & proposed changes
General idea

IPP Client

authorization procedure

access token

IPP requests with access token

Authorization Server

IPP Printer
Possible configurations
Main Assumptions

- **IPP Printer** can be managed by only one **Authorization Server**
- **IPP Printer** knows the URL of its **Authorization Server**
- **IPP Client** does not need any prior knowledge about the implementation of **IPP Printer** or **Authorization Server**
- **IPP Printer** does not need any prior knowledge about the implementation of **IPP Client**
- All communication between **IPP Client** and **IPP Printer** and between **IPP Client** and **Authorization Server** relies on https protocol
Out of Scope

- Communication between IPP Printer and Authorization Server
  - Verification of the access token performed by IPP Printer
- Capabilities of IPP Printer and the way jobs are processed
  - IPP version supported by IPP Printer
  - Printing pipeline - filters needed to process the document
- Source of knowledge of IPP Printers
  - Provided by user
  - Queried from Authorization Server or printing server
  - Discovered via mDNS
Security considerations

1. Communication between IPP Client and IPP Printer cannot be intercepted by any third party.
   The immediate goal: to protect user data.

2. Access to IPP Printer can be restricted to a limited set of authorized users.
   The immediate goal: to protect printer resources (e.g., paper, ink, printing time, etc.).

The second condition may be achieved only if the first requirement is fulfilled. Otherwise, attackers would be able to intercept credentials/access tokens and impersonate authorized users.
Mitigating possible attacks - fake Authorization Server

Both requirements must be fulfilled:

1. **Authorization Server** must have a valid certificate that is fully verified by the IPP Client
2. The URL of the **Authorization Server** must be trusted
   - Possible sources of **Authorization Server** URLs:
     - Well-known FQDN of the service
     - Provided by the administrator of the system/local network
     - Provided by the user
     - Provided by the IPP Printer
       - Must be explicitly verified by the user!
Mitigating possible attacks - fake IPP Printer

Both requirements must be fulfilled:

1. IPP Printer must have a valid certificate that is fully verified by the IPP Client

2. The Authorization Server must verify the identity of the IPP Printer
   - Possible approaches to identity verification
     - IPP Printer has FQDN that can be verified by the Authorization Server
     - Authorization Server verifies the fingerprint of the IPP Printer’s certificate
       - An alternative for printers visible only in local network and without unique addresses (e.g., discovered via mDNS)
Proposed protocol

IPP Client

Client

Internet Browser

Authorization Request
(client_id, scope)
(authorization code)

Token Request
(authorization code)
(access token)

Token Exchange Request
(access token, IPP Printer URI)
(endpoint access token)

IPP Request
(endpoint access token)

Authorization Server

Authorization Endpoint

Token Endpoint

IPP Printer

Google Open Source
Proposed protocol

1. **IPP Printer** managed by **Authorization Server** MUST return attributes:
   a. `oauth-authorization-server-uri` (always)
   b. `oauth-authorization-scope` (if needed).

2. **IPP Client** MUST:
   a. check that `oauth-authorization-server-uri` is on the list of trusted servers
   b. query metadata from the **Authorization Server** as described in RFC 8414
   c. try to register as a new client as described in RFC 7591 when:
      i. `client_id` is not known, AND
      ii. the **Authorization Server** allows for dynamic registration of new clients.
Proposed protocol

1. **IPP Client** MUST open session with **Authorization Server** as described in RFC 6749:
   a. the **IPP Client** uses an internet browser to open authorization link from **Authorization Server** and enables the user to complete authentication procedure provided by the server;
   b. the **IPP Client** obtains *access token* (and, if provided, *refresh token*) from the **Authorization Server**

2. The **IPP Client** uses *access token* to obtain *endpoint access token* for specific **IPP Printer** as described in RFC 8693
   a. the **IPP Client** sends to the **Authorization Server** the URL of the **IPP Printer** and the fingerprint of its certificate
Implementation Plans

- **IPP Client** in ChromeOS
  - experimental feature
  - activated by a flag

- Convince our partners to implement **Authorization Server** on their side
  - centralized solutions with infrastructure printers

- Future: stand-alone **Authorization Server** working with **IPP Printer** being:
  - print server - requires protocol between **IPP Printer** and **Authorization Server**
  - stand-alone printer - as above + OEM that agree to implement the protocols
Proposed changes

- **IPP Printer** should announce `oauth-authorization-server-uri` and `-scope` in HTTP header
  - Access to Get-Printer-Attributes request can be restricted too
  - Get-Printer-Attributes may be used to conduct DDOS attack

- Provide standard way of querying list of **IPP Printers** from the **Authorization Server**
  - It may be useful for some configurations

- **IPP Client** should be able to delegate to **Authorization Server** verification of a certificate of **IPP Printer**
  - **IPP Client** would not need additional configuration to verify **IPP Printer's** certificate
Thank you!

Benjamin Gordon  
Software Engineer  
bmgordon@chromium.org

Piotr Pawliczkek  
Software Engineer  
pawliczek@chromium.org

Google Open Source