Stream:	Internet Engineering Task Force (IETF)
RFC:	10027
BCP:	247
Category:	Best Current Practice
Published:	August 2026
ISSN:	2070-1721
Authors:	P. Kasselman	D. Fett	F. Skokan
---	---	---	---
	Defakto Security	*Authlete*	*Okta*

RFC 10027
Best Current Practice for Security of Cross-Device Flows

Abstract

This document describes threats against cross-device flows along with practical mitigations, protocol selection guidance, and a summary of formal analysis results identified as relevant to the security of cross-device flows. It serves as a security guide to system designers, architects, product managers, security specialists, fraud analysts, and engineers implementing cross-device flows.

Status of This Memo

This memo documents an Internet Best Current Practice.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on BCPs is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc10027.

Copyright Notice

Table of Contents

1. Introduction

Protocol flows that span multiple end-user devices are in widespread use today. These flows are often referred to as cross-device flows. A common example is a user that uses their mobile phone to scan a QR code from their smart TV, giving an app on the TV access to their video streaming service. Besides QR codes, other mechanisms are often used, such as PIN codes that the user has to enter on one of the devices or push notifications to a mobile app that the user has to approve.

In all cases, it is up to the user to decide whether or not to grant authorization. However, the QR code or PIN is transferred via an unauthenticated channel, leaving it up to the user to decide in which context an authorization is requested. This may be exploited by attackers to gain unauthorized access to a user's resources.

To accommodate the various nuances of cross-device flows, this document distinguishes between use cases where the cross-device flow is used to authorize access to a resource (cross-device authorization flows) and use cases where the cross-device flow is used to transfer an existing session (cross-device session transfer flows).

1.1. Cross-Device Authorization

Cross-device authorization flows enable a user to initiate an authorization flow on one device (the Consumption Device) and then use a second, personally trusted, device (the Authorization Device) to authorize the Consumption Device to access a resource (e.g., access to a service). The device authorization grant [RFC8628] and Client-Initiated Backchannel Authentication (CIBA) [CIBA] are two examples of popular cross-device authorization flows.

In these flows, the Consumption Device and the Authorization Device are not directly connected, and there are no technical mechanisms for the Authorization Device and Consumption Device to establish mutual authentication. It is left to the user to decide whether the source of the authorization request (the Consumption Device) should be trusted before they scan a QR code, enter a user code, or accept an authorization request pushed to their Authorization Device. The transfer of the authorization request and context between the Consumption Device and Authorization Device is done over an unauthenticated channel. The only mitigation against this unauthenticated channel is the user's judgment.

Cross-Device Consent Phishing (CDCP) attacks exploit the unauthenticated channel between the Consumption Device and Authorization Device using social engineering techniques commonly used in phishing attacks to gain unauthorized access to the user's data.

Several publications have emerged in the public domain ([ARTDCPHISH], [DCFLOWPHISH], [NEWDCPHISH], [DEFCON29], [DCATTACK], and [SQPHISH]) that describe how the unauthenticated channel can be exploited using social engineering techniques borrowed from phishing. Unlike conventional phishing attacks, these attacks don't harvest credentials. Instead, they skip the step of collecting credentials by persuading users to grant authorization using their Authorization Devices.

Once the user grants authorization, the attacker has access to the user's resources and in some cases is able to collect access and refresh tokens. Once in possession of the access and refresh tokens, the attacker may use these tokens to execute lateral attacks and gain additional access, or monetize the tokens by selling them. These attacks are effective even when multi-factor authentication is deployed, since the attacker's aim is not to capture and replay the credentials, but rather to persuade the user to grant authorization.

1.2. Cross-Device Session Transfer

Session transfer flows enable a user to transfer access to a service or network from a device on which the user is already authenticated to a second device such as a mobile phone. In these flows, the user is authenticated and then authorizes the session transfer on one device, referred to as the Authorization Device (e.g., a personal computer, web portal, or application), and transfers the session to the device where they will continue to consume the session, referred to as the Consumption Device (e.g., a mobile phone or portable device).

The session may be transferred by showing the user a session transfer code on the Authorization Device, which is then entered on the Consumption Device. This flow may be streamlined by rendering the session transfer code as a QR code on the Authorization Device and scanning it with the Consumption Device.

The session transfer preserves state information, including authentication state, at the second device to avoid additional configuration and optimize the user experience. These flows are often used to add new devices to a network, onboard customers to a mobile application, or provision new credentials (e.g., as described in [OpenID.SIOPv2]).

In these cross-device session transfer flows, the channel between the Authorization Device and the Consumption Device is unauthenticated.

Cross-Device Session Phishing (CDSP) attacks exploit the unauthenticated channel between the Authorization Device and Consumption Device by using social engineering techniques to convince the user to send the session transfer code to the attacker. These attacks borrow techniques from conventional phishing attacks, but instead of collecting passwords, attackers collect session transfer codes and other artifacts that allow them to set up a session and then use it to access a user's data.

1.3. Defending Against Cross-Device Attacks

This document provides guidance to implementers (e.g., system designers, architects, product managers, security specialists, fraud analysts, and engineers) of cross-device flows to defend against CDCP and CDSP attacks. This guidance includes:

1. Practical mitigations for susceptible protocols (Section 6.1).
2. Protocol selection guidance to avoid using susceptible protocols (Section 6.2).
3. Results from formal analysis of susceptible protocols (Section 6.3).

1.4. Conventions and Terminology

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This specification uses the terms "access token", "refresh token", "authorization server", "resource server", "authorization request", and "client" defined in "The OAuth 2.0 Authorization Framework" [RFC6749].

This document uses the terms "social engineering" and "phishing" as described in the NIST Computer Security Resource Center Glossary [NISTGlossary].

2. Best Practices

This section describes the set of security mechanisms and measures to secure cross-device protocols against CDCP and CDSP attacks that the OAuth Working Group considers best practices at the time of writing this specification.

1. Implementers **MUST** perform a risk assessment before implementing cross-device flows, weighing the risks from CDCP and CDSP attacks against benefits for users.
2. Implementers **SHOULD** avoid cross-device flows if risks cannot be sufficiently mitigated.
3. Implementers **SHOULD** follow the guidance provided in Section 6.2 for protocol selection.
4. Implementers **MUST** select appropriate mitigations from Section 6.1 to address risks identified in the risk assessment.
5. Implementers **SHOULD** include proximity as one of the selected mitigations as defined in Section 6.1.1, if possible.

These best practices apply to the device authorization grant [RFC8628] as well as other cross-device protocols such as CIBA [CIBA], Self-Issued OpenID Provider v2 [OpenID.SIOPv2], OpenID for Verifiable Presentations [OpenID.VP], the Pre-Authorized Code Flow in [OpenID.VCI], and other cross-device protocols that rely on the user to authenticate the channel between devices.

Section 3 provides details about susceptible protocols, and Section 4 provides attack descriptions. Section 5 provides an overview of existing protocols and standards. Section 6.1 provides details about the security mechanisms and mitigations, Section 6.2 provides protocol selection guidance, and Section 6.3 provides details from formal analysis of protocols that apply to cross-device flows.

3. Cross-Device Flow Patterns

Cross-device flows enable a user to initiate an interaction on one device (e.g., a smart TV) and complete or continue it on a second device (e.g., a mobile phone). This specification addresses two distinct cross-device use cases:

- **Cross-Device Authorization:** In the cross-device authorization use case, the second device is used to authenticate the user or grant authorization before passing control back to the first device as described in Section 3.1.
- **Cross-Device Session Transfer:** In the cross-device session transfer use case, the user is already authenticated on the first device before the session is transferred to the second device without requiring the user to re-authenticate as described in Section 3.2.

These flows typically involve using a mobile phone to scan a QR code or enter a user code displayed on the first device (e.g., a smart TV, kiosk, personal computer, or other electronic device).

3.1. Cross-Device Authorization

In a cross-device authorization flow, a user attempts to access a service on one device, referred to as the Consumption Device (e.g., a smart TV), and then uses a second device, referred to as the Authorization Device (e.g., a smartphone), to authorize access to a resource (e.g., access to a streaming service) on the Consumption Device.

Cross-device authorization flows have several benefits, including:

- Authorization on devices with limited input capabilities: End users can authorize devices with limited input capabilities to access content (e.g., smart TVs, digital whiteboards, printers, or similarly constrained devices).
- Secure authentication on shared or public devices: End users can perform authentication and authorization using a personally trusted device, without risk of disclosing their credentials to a public or shared device.
- Ubiquitous multi-factor authentication: Enables a user to use multi-factor authentication, independent of the device on which the service is being accessed (e.g., a kiosk, smart TV, or shared personal computer).
- Convenience of a single, portable, credential store: Users can keep all their credentials in a mobile wallet or mobile phone that they already carry with them.

There are three cross-device flow patterns for transferring the authorization request between the Consumption Device and the Authorization Device.

- **User-Transferred Session Data Pattern:** In this pattern, the user initiates the authorization process with the Authorization Server by copying information from the Consumption Device to the Authorization Device, before authorizing an action. By transferring the data from the Consumption Device to the Authorization Device, the user transfers the authorization

session. For example, the user may read a code displayed on the Consumption Device and enter it on the Authorization Device, or they may scan a QR code displayed on the Consumption Device with the Authorization Device. The device authorization grant [RFC8628] is an example of a cross-device flow that follows this pattern.

- **Backchannel-Transferred Session Pattern:** In this pattern, the OAuth client on the Consumption Device is responsible for transferring the session and initiating authorization on the Authorization Device via a backchannel with the Authorization Server. For example, the user may attempt an online purchase on a Consumption Device (e.g., a personal computer) and receive an authorization request on their Authorization Device (e.g., a mobile phone). CIBA [CIBA] is an example of a cross-device flow that follows this pattern.
- **User-Transferred Authorization Data Pattern:** In this pattern, the OAuth client on the Consumption Device triggers the authorization request via a backchannel with the Authorization Server. Authorization data (e.g., a 6-digit authorization code) is displayed on the Authorization Device, which the user transfers to the Consumption Device (e.g., by manually entering it). For example, the user may attempt to access data in an enterprise application and receive a 6-digit authorization code on their Authorization Device (e.g., a mobile phone) that they enter on the Consumption Device. Note that the use of a 6-digit code is illustrative and reflects common practice at the time of writing. Code length may vary based on usability and risk considerations, and specifying the appropriate length is out of scope for this document.

3.1.1. User-Transferred Session Data Pattern

The device authorization grant [RFC8628] is an example of a cross-device flow that relies on the user copying information from the Consumption Device to the Authorization Device by either entering data manually or scanning a QR code. Figure 1 shows a typical example of this flow.

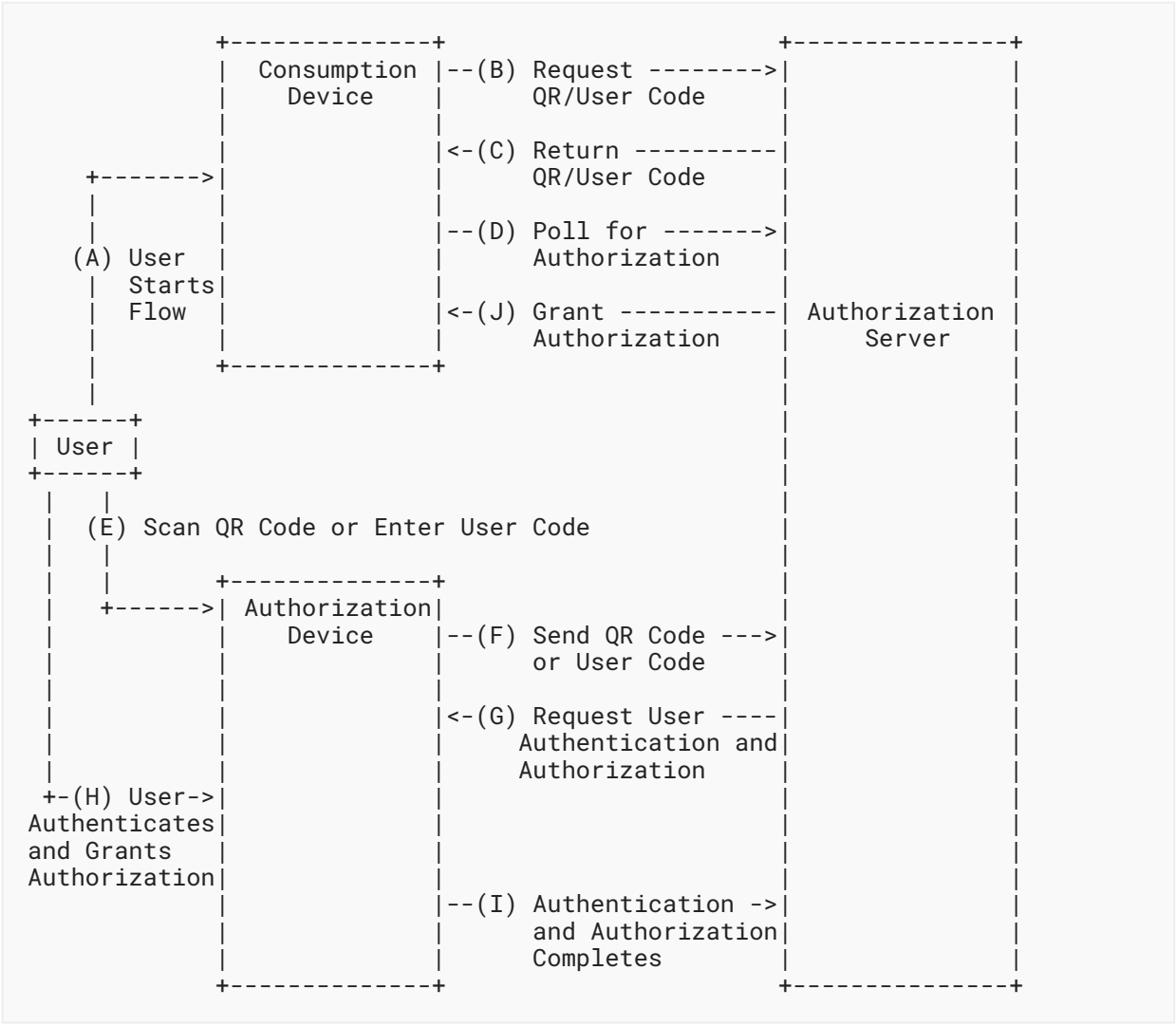

Figure 1: User-Transferred Session Data Pattern

(A) The user takes an action on the Consumption Device by starting a purchase, adding a device to a network, or connecting a service to the Consumption Device.

(B) The Consumption Device requests a QR code or user code from an Authorization Server.

(C) The Authorization Server returns a QR code or user code to the Consumption Device, which displays it to the user with instructions to scan the QR code or enter the user code using the Authorization Device.

(D) The Consumption Device starts polling the Authorization Server to find out if the user granted authorization.

(E) The user scans the QR code or enters the user code on the Authorization Device.

(F) The QR code or user code is sent to the Authorization Server.

(G) The Authorization Server validates the QR code or user code and prompts the user to authenticate and either accept or decline the authorization request.

(H) The user authenticates and grants authorization using the Authorization Device.

(I) The user is authenticated, and authorization is granted to access the user's resources (there may be several additional messages, depending on the authentication protocol, user interface, and other implementation details).

(J) The Authorization Server grants authorization (e.g., by issuing tokens) to the Consumption Device to access the user's resources.

3.1.2. Backchannel-Transferred Session Pattern

CIBA [CIBA] transfers the session on the backchannel with the Authorization Server to request authorization on the Authorization Device. Figure 2 shows an example of this flow.

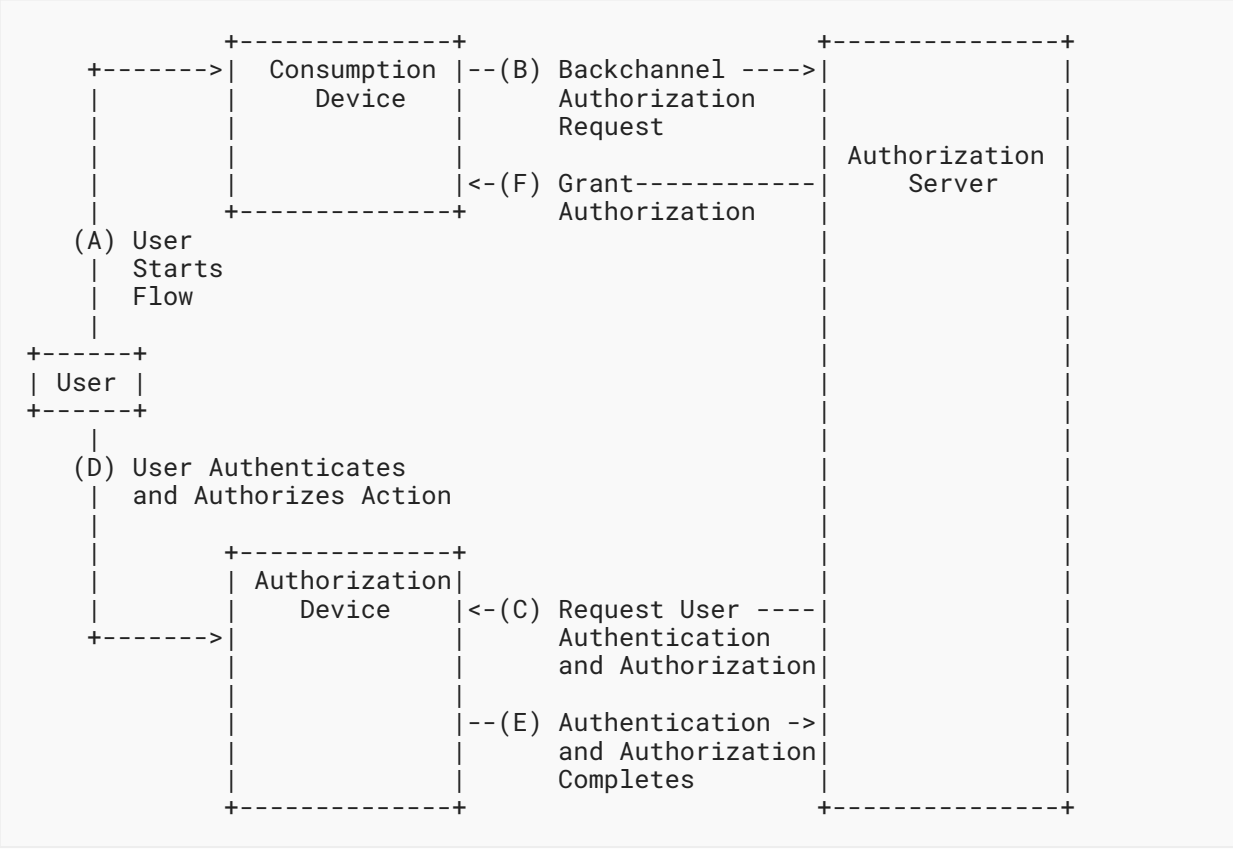

Figure 2: Backchannel-Transferred Session Pattern

(A) The user takes an action on the Consumption Device by starting a purchase, adding a device to a network, or connecting a service to the Consumption Device.

(B) The client on the Consumption Device requests user authorization on the backchannel from the Authorization Server, instructs the user to authorize the request on the Authorization Device, and waits for a response from the Authorization Server.

(C) The Authorization Server requests user authentication and authorization on the user's Authorization Device.

(D) If the user is unauthenticated, they use their Authorization Device to authenticate and grant authorization to the Authorization Server.

(E) The user is authenticated, and authorization is granted to access the user's resources (there may be several additional messages, depending on the authentication protocol, user interface, and other implementation details).

(F) The Authorization Server grants authorization (e.g., by issuing tokens) to the Consumption Device to access the user's resources.

The Authorization Server may use a variety of mechanisms to request user authorization, including a push notification to a dedicated app on a mobile phone or sending a text message with a link to an endpoint where the user can authenticate and authorize an action.

3.1.3. User-Transferred Authorization Data Pattern

Examples of the User-Transferred Authorization Data Pattern include flows in which the Consumption Device requests the Authorization Server to send authorization data (e.g., a 6-digit authorization code in a text message, email, or mobile application) to the Authorization Device. Once the Authorization Device receives the authorization data, the user enters it on the Consumption Device. The Consumption Device sends the authorization data back to the Authorization Server for validation before gaining access to the user's resources. Figure 3 shows an example of this flow.

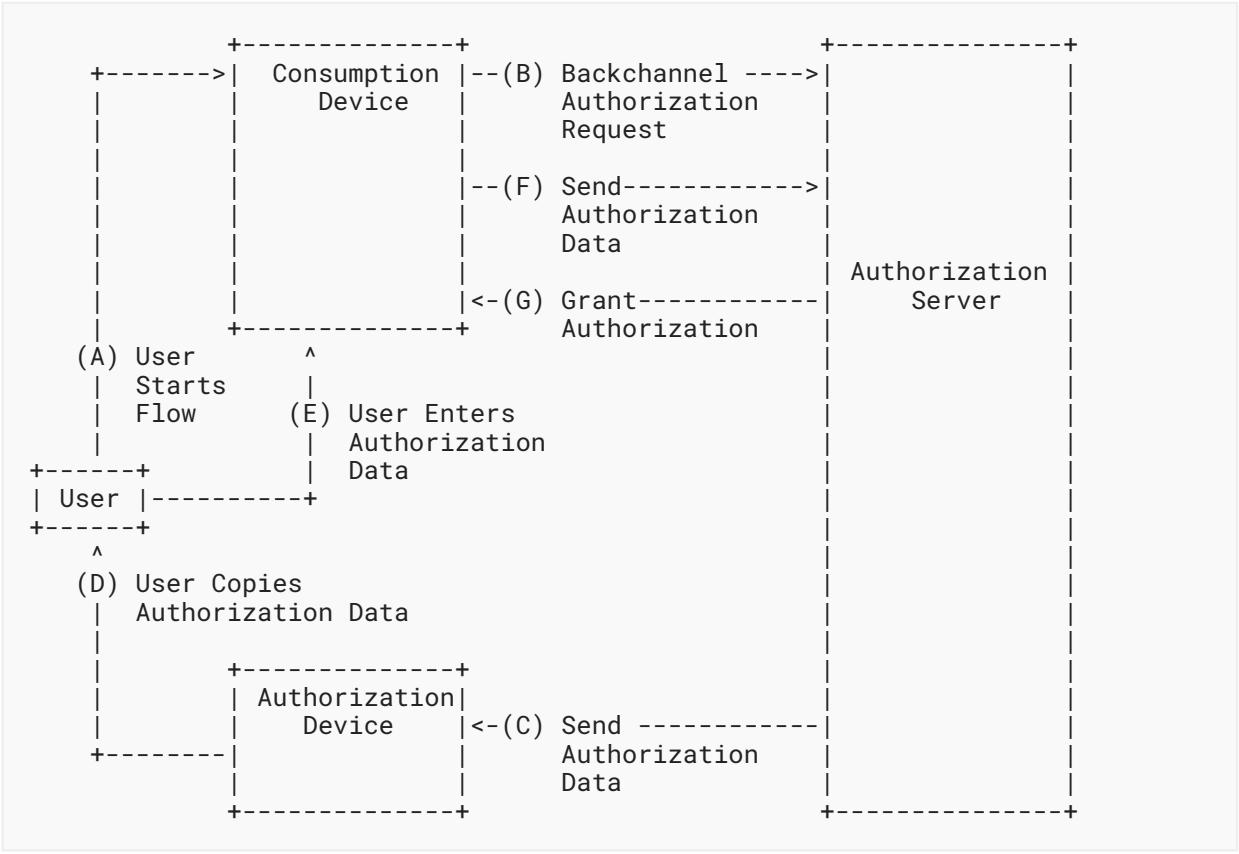

Figure 3: User-Transferred Authorization Data Pattern

(A) The user takes an action on the Consumption Device by starting a purchase, adding a device to a network, or connecting a service to the Consumption Device.

(B) The client on the Consumption Device requests user authorization on the backchannel from the Authorization Server.

(C) The Authorization Server sends authorization data (e.g., a 6-digit authorization code) to the Authorization Device. Examples of mechanisms that may be used to distribute the authorization data include text messages, email, or a mobile application.

(D) The user reads and copies the authorization data (e.g., the 6-digit authorization code) received on the Authorization Device.

(E) The user enters the authorization data on the Consumption Device.

(F) The Consumption Device sends the authorization data to the Authorization Server.

(G) The Authorization Server issues tokens or grants authorization to the Consumption Device to access the user's resources if the authorization data is the same as that sent in step (C).

The Authorization Server may choose to authenticate the user before sending the authorization data.

3.2. Cross-Device Session Transfer

Session transfer flows enable a user to transfer access to a service or network from a device on which the user is already authenticated to a second device such as a mobile phone. In these flows, the user is authenticated and then authorizes the session transfer on one device, referred to as the Authorization Device (e.g., a personal computer, web portal or application), and transfers the session to the device where they will continue to consume the session, referred to as the Consumption Device (e.g., a mobile phone or portable device).

The session transfer preserves state information, including authentication state, at the second device to avoid additional configuration and optimize the user experience. These flows are often used to add new devices to a network, onboard customers to a mobile application, or provision new credentials (e.g., [OpenID.SIOPv2]).

3.2.1. Cross-Device Session Transfer Pattern

In this flow, the user is authenticated and starts the flow by authorizing the transfer of the session on the Authorization Device. The Authorization Device requests a session transfer code, which may be rendered as a QR code on the Authorization Device. When the user scans the QR code or enters it on the Consumption Device where they would like the session to continue, the Consumption Device presents it to the Authorization Server. The Authorization Server then transfers the session to the Consumption Device. This may include transferring authentication and authorization state to optimize the user experience. This type of flow is used, for example, for adding new devices to networks, bootstrapping new applications, or provisioning new credentials. The Pre-Authorized Code Flow in [OpenID.VCI] is an instance of using this pattern to provision a new credential. Figure 4 shows a typical flow.

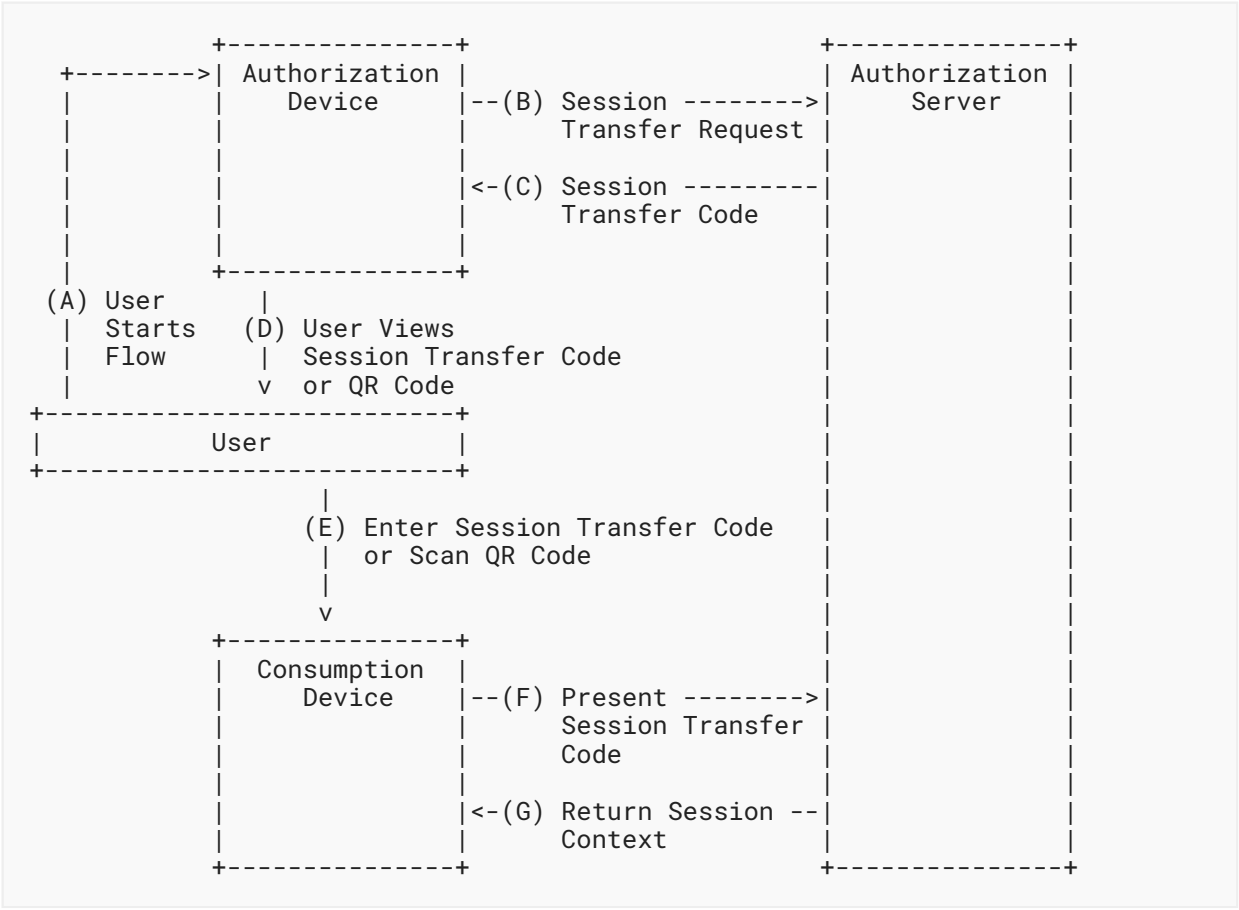

Figure 4: Cross-Device Session Transfer Pattern

(A) The user is authenticated on the Authorization Device and authorizes the transfer of the session to the Consumption Device.

(B) The user starts the flow and is authenticated on their Authorization Device before they authorize the transfer of the session to the Consumption Device.

(C) The Authorization Server responds with a session transfer code, which may be rendered as a QR code on the Authorization Device.

(D) The user views the session transfer code, which may be rendered as a QR code.

(E) The user enters the session transfer code on the Consumption Device (e.g., their mobile phone). If the session transfer code is rendered as a QR code, the user scans the QR code with the target Consumption Device.

(F) The client on the Consumption Device presents the session transfer code to the Authorization Server.

(G) The Authorization Server verifies the session transfer code and returns the session context information needed to resume the session on the Consumption Device. The user resumes the session they initiated and authorized on the Authorization Device and proceeds to access the information on the Consumption Device.

3.3. Examples of Cross-Device Flows

The following examples illustrate the above flows in a diverse range of practical settings. Corresponding examples of how these flows may be exploited by attackers are documented in Section 4.3.

3.3.1. Example A1: Authorize Access to a Video Streaming Service (User-Transferred Session Data Pattern)

An end user sets up a new smart TV and wants to connect it to their favorite streaming service. The streaming service displays a QR code on the TV that the user scans with their mobile phone. The user is redirected to the streaming service provider's web page and asked to enter their credentials to authorize the smart TV to access the streaming service. The user enters their credentials and grants authorization, after which the streaming service is available on the smart TV. Section 4.3.1 illustrates an exploit that applies to this scenario.

3.3.2. Example A2: Authorize Access to Productivity Services (User-Transferred Session Data Pattern)

An employee wants to access their files on an interactive whiteboard in a conference room. The interactive whiteboard displays a URL and a code. The user enters the URL on their personal computer and is prompted for the code. Once they enter the code, the user is asked to authenticate and authorize the interactive whiteboard to access their files. The user enters their credentials and authorizes the transaction, and the interactive whiteboard retrieves their files and allows the user to interact with the content. Section 4.3.2 describes an exploit relevant to this example.

3.3.3. Example A3: Authorize Use of a Bike Sharing Scheme (User-Transferred Session Data Pattern)

An end user wants to rent a bicycle from a bike sharing scheme. The bicycles are locked in bicycle racks on sidewalks throughout a city. To unlock and use a bicycle, the user scans a QR code on the bicycle using their mobile phone. Scanning the QR code redirects the user to the bicycle sharing scheme's authorization page where the user authenticates and authorizes payment for renting the bicycle. Once authorized, the bicycle sharing service unlocks the bicycle, allowing the user to use it to cycle around the city. Section 4.3.3 outlines an exploit relevant to this situation.

3.3.4. Example A4: Authorize a Financial Transaction (Backchannel-Transferred Session Pattern)

An end user makes an online purchase. Before completing the purchase, they get a notification on their mobile phone, asking them to authorize the transaction. The user opens their app and authenticates to the service before authorizing the transaction. Section 4.3.4 describes two exploits relevant to this example.

3.3.5. Example A5: Add a Device to a Network (Cross-Device Session Transfer Pattern)

An employee is issued a personal computer that is already joined to a network. The employee wants to add their mobile phone to the network to allow it to access corporate data and services (e.g., files and email). The employee is logged in on the personal computer where they initiate the process of adding their mobile phone to the network. The personal computer displays a QR code that authorizes the user to join their mobile phone to the network. The employee scans the QR code with their mobile phone, and the mobile phone is joined to the network. The employee can start accessing corporate data and services on their mobile device. Section 4.3.5 gives an example of how this flow may be exploited.

3.3.6. Example A6: Remote Onboarding (User-Transferred Session Data Pattern)

A new employee is directed to an onboarding portal to provide additional information to confirm their identity on their first day with their new employer. Before activating the employee's account, the onboarding portal requests that the employee present a government issued ID, proof of a background check, and proof of their qualifications. The onboarding portal displays a QR code, which the user scans with their mobile phone. Scanning the QR code invokes the employee's digital wallet on their mobile phone, and the employee is asked to present digital versions of an identity document (e.g., a driving license), proof of a background check by an identity verifier, and proof of their qualifications. The employee authorizes the release of the credentials, and after completing the onboarding process, their account is activated. Section 4.3.6 provides an example of an exploit for this use case.

3.3.7. Example A7: Application Bootstrap (Cross-Device Session Transfer Pattern)

An employee is signed in to an application on their personal computer and wants to bootstrap the mobile application on their mobile phone. The employee initiates the cross-device flow and is shown a QR code in their application. The employee launches the mobile application on their phone and scans the QR code, which results in the user being signed in to the application on the mobile phone. Section 4.3.7 describes an exploit that applies to this scenario.

3.3.8. Example A8: Access a Productivity Application (User-Transferred Authorization Data Pattern)

A user is accessing a Computer-Aided Design (CAD) application. When accessing the application, authorization data in the form of a 6-digit authorization code is sent to the user's mobile phone. The user views the 6-digit authorization code on their phone and enters it in the CAD application, after which the CAD application displays the user's most recent designs. Section 4.3.8 outlines an attack relevant to this scenario.

3.3.9. Example A9: Administer a System (Backchannel-Transferred Session Pattern)

A network administrator wants to access an administration portal used to configure network assets and deploy new applications. When attempting to access the service, the network administrator receives a notification in an app on their mobile device, requesting them to confirm access to the portal. The network administrator approves the request on their mobile phone and is granted access to the portal. Section 4.3.9 describes how an attacker might exploit this flow.

4. Cross-Device Flow Exploits

Attackers exploit the absence of an authenticated channel between the two devices used in a cross-device flow by using social engineering techniques typically used in phishing attacks.

In cross-device authorization flows, the attacker uses these social engineering techniques by changing the context in which the authorization request is presented to convince the user to grant authorization when they shouldn't. These attacks are also known as CDCP attacks.

In cross-device session transfer flows, the attacker uses these social engineering techniques to convince the user to initiate a session transfer and send them a session transfer code. Once the attacker is in possession of this session transfer code, they present it to the Authorization Server to transfer the session and access the user's resources. These attacks are referred to as CDSP attacks.

4.1. Cross-Device Authorization Flow Exploits

Attackers exploit cross-device authorization flows by initiating an authorization flow on the Consumption Device and then use social engineering techniques to change the context in which the request is presented to the user in order to convince them to grant authorization on the Authorization Device. The attacker is able to change the context of the authorization request because the channel between the Consumption Device and the Authorization Device is unauthenticated. These attacks are also known as CDCP attacks.

4.1.1. User-Transferred Session Data Pattern Exploits

A common action in cross-device flows is to present the user with a QR code or a user code on the Consumption Device (e.g., smart TV), which is then scanned or entered on the Authorization Device (e.g., mobile phone). When the user scans the code or copies the user code, they do so without any proof that the QR code or user code is being displayed in the place or context intended by the service provider. It is up to the user to decide whether they should trust the QR code or user code. In effect, the user is asked to compensate for the absence of an authenticated channel between the Consumption Device (e.g., smart TV) and the Authorization Device (e.g., mobile phone).

Attackers exploit this absence of an authenticated channel between the two devices by obtaining QR codes or user codes (e.g., by initiating the authorization flows). They then use social engineering techniques to change the context in which authorization is requested to convince

end users to scan the QR code or enter it on their Authorization Device (e.g., mobile phone). Once the end user performs the authorization on the mobile device, the attacker who initiated the authentication or authorization request obtains access to the user's resources. Figure 5 shows an example of such an attack.

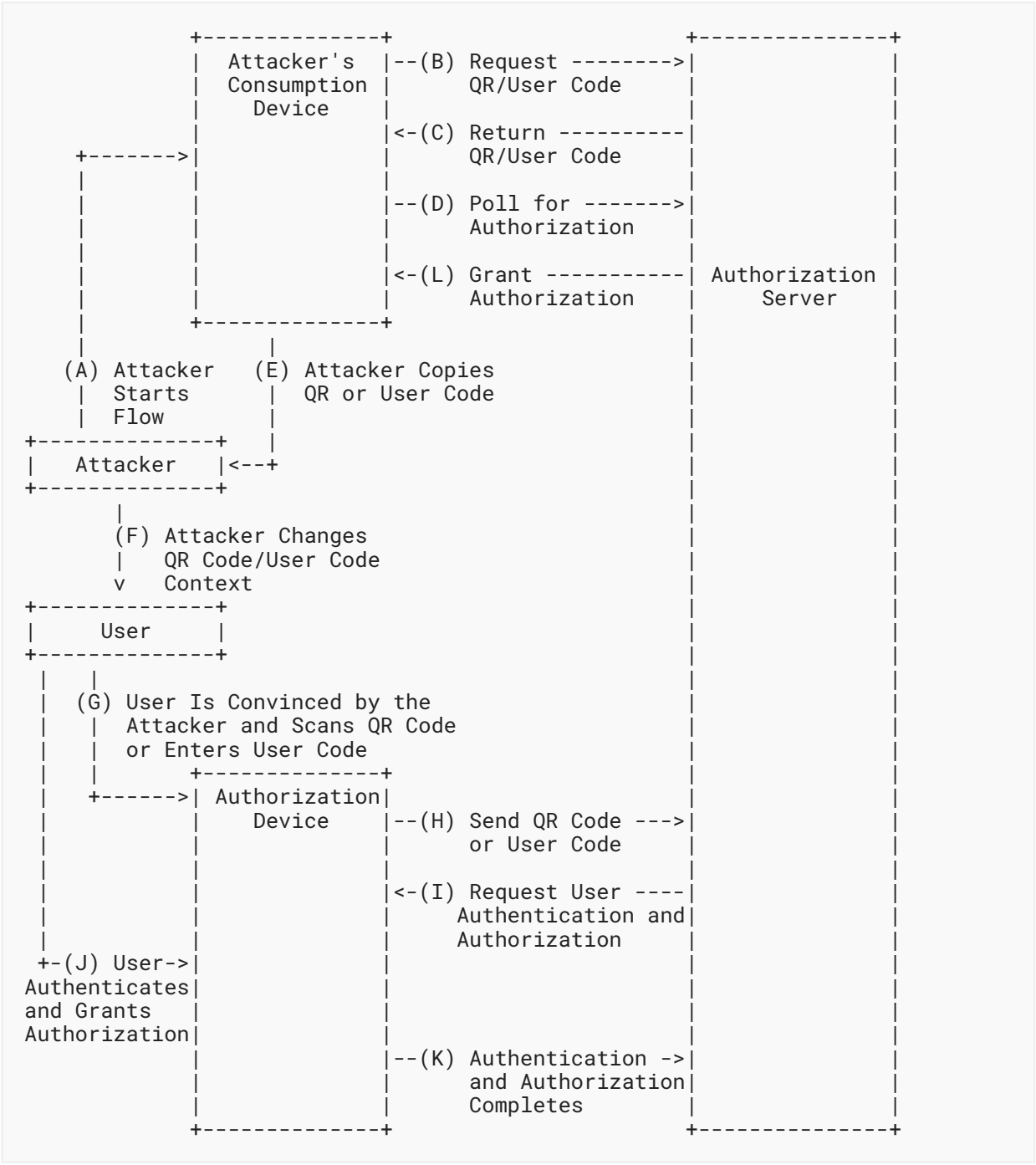

Figure 5: User-Transferred Session Data Pattern Exploit

(A) The attacker initiates the protocol on the Consumption Device (or mimics the Consumption Device) by starting a purchase, adding a device to a network, or connecting a service to the Consumption Device.

(B) The Consumption Device requests a QR code or user code from an Authorization Server.

(C) The Authorization Server returns a QR code or user code to the Consumption Device, which displays it to the user with instructions to scan the QR code or enter the user code using the Authorization Device.

(D) The Consumption Device starts polling the Authorization Server to find out if the user granted authorization.

(E) The attacker copies the QR code or user code.

(F) The attacker changes the context in which the QR code or user code is displayed in such a way that the user is likely to scan the QR code or use the user code when completing the authorization. For example, the attacker could craft an email that includes the user code or QR code and send it to the user. The email might encourage the user to scan the QR code or enter the user code by suggesting that doing so would grant them a reward through a loyalty program or prevent the loss of their data.

(G) The QR code or user code is displayed to the user in a context chosen by the attacker. The user is convinced by the attacker's effort and scans the QR code or enters the user code on the Authorization Device.

(H) The QR code or user code is sent to the Authorization Server.

(I) The Authorization Server validates the QR code or user code and prompts the user to authenticate and accept or decline the authorization request.

(J) The user authenticates and grants authorization using the Authorization Device.

(K) The user is authenticated with the Authorization Server, and authorization is granted to access the user's resources (there may be several additional messages, depending on the authentication protocol, user interface, and other implementation details).

(L) The Authorization Server issues tokens or grants authorization to the Consumption Device, which is under the attacker's control, to access the user's resources. The attacker gains access to the resources and any authorization artifacts like access and refresh tokens.

4.1.2. Backchannel-Transferred Session Pattern Exploits

In the Backchannel-Transferred Session Pattern, the client requests the Authorization Server to authenticate the user and obtain authorization for an action. This may happen as a result of user interaction with the Consumption Device but may also be triggered without the user's direct interaction with the Consumption Device, resulting in an authorization request presented to the user without context of why or who triggered the request.

Attackers exploit this lack of context by using social engineering techniques to prime the user for an authorization request and thereby convince them to grant authorization. The social engineering techniques range in sophistication from messages misrepresenting the reason for

receiving an authorization request to triggering a large volume of requests at an inconvenient time for the user, in the hope that the user will grant authorization to make the requests stop. Figure 6 shows an example of such an attack.

The ability to trigger authorization requests without user involvement can be exploited by an attacker to overwhelm users with a high volume of requests in a short period, increasing the likelihood of inadvertent approval.

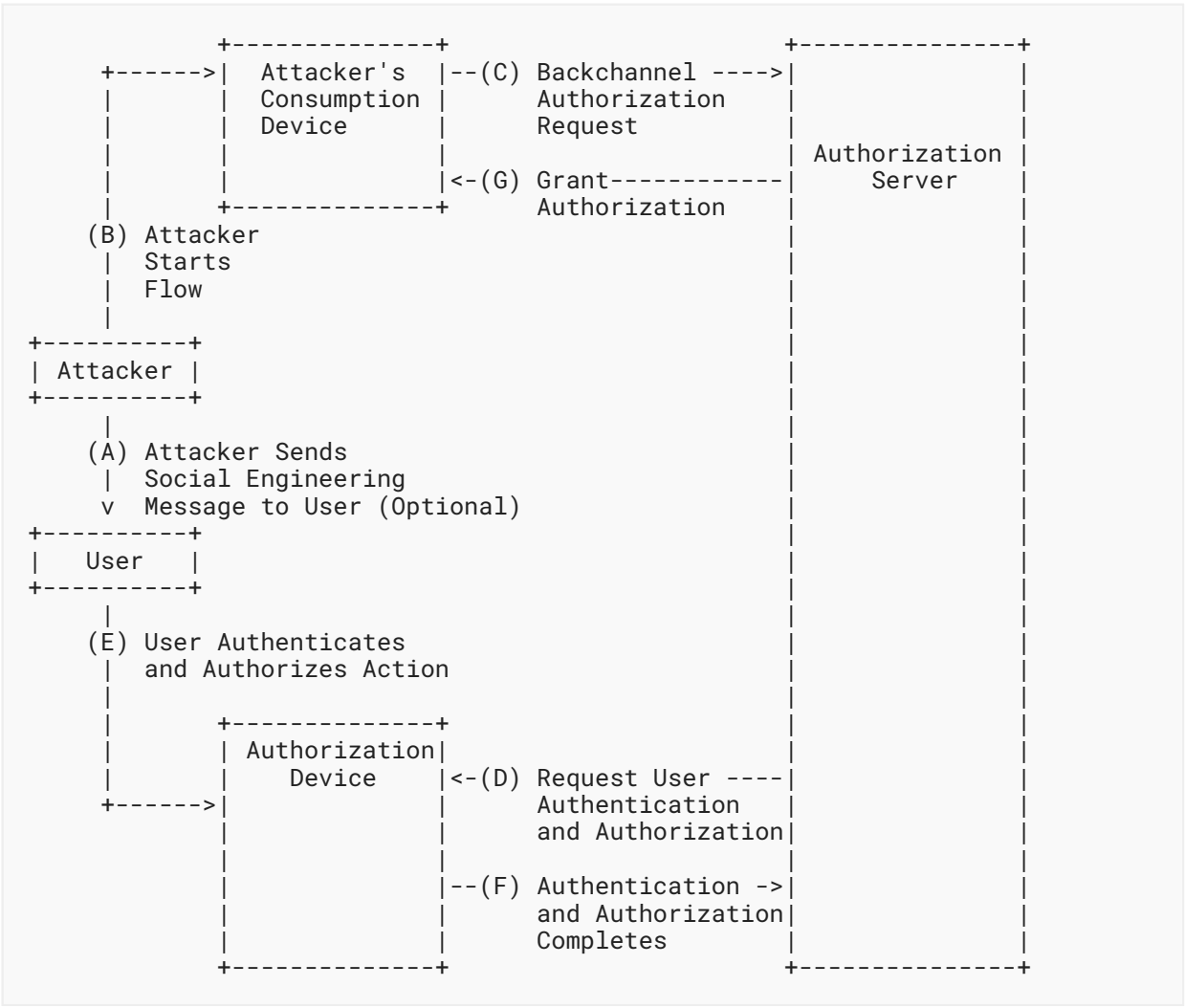

Figure 6: Backchannel-Transferred Session Pattern Exploit

(A) The attacker sends a social engineering message to prepare the user for the upcoming authorization (optional).

(B) The attacker initiates the protocol on the Consumption Device (or mimics the Consumption Device) by starting a purchase, adding a device to a network, or accessing a service on the Consumption Device.

(C) The client on the Consumption Device requests user authorization on the backchannel from the Authorization Server and waits for a response from the Authorization Server.

(D) The Authorization Server requests user authentication and authorization on the user's Authorization Device.

(E) If the user is unauthenticated, they use their Authorization Device to authenticate and grant authorization to the Authorization Server.

(F) The user is authenticated, and authorization is granted to access the user's resources (there may be several additional messages, depending on the authentication protocol, user interface, and other implementation details).

(G) The Authorization Server issues tokens or grants authorization to the Consumption Device, which is under the attacker's control. The attacker gains access to the user's resources and possibly any authorization artifacts like access and refresh tokens.

4.1.3. User-Transferred Authorization Data Pattern Exploits

In cross-device flows that follow the User-Transferred Authorization Data Pattern, the client on the Consumption Device initiates the authorization request, but the user still has to transfer the authorization data to the Consumption Device. The authorization data may take different forms, including a numerical value such as a 6-digit authorization code. The authorization request may happen as a result of user interaction with the Consumption Device but may also be triggered without the user's direct interaction with the Consumption Device.

Attackers exploit the User-Transferred Authorization Data Pattern by combining the social engineering techniques used to set context for users and convincing users to provide them with authorization data sent to their Authorization Devices (e.g., mobile phones). These attacks are very similar to phishing attacks, except that the attacker also has the ability to trigger the authorization request to be sent to the user directly by the Authorization Server.

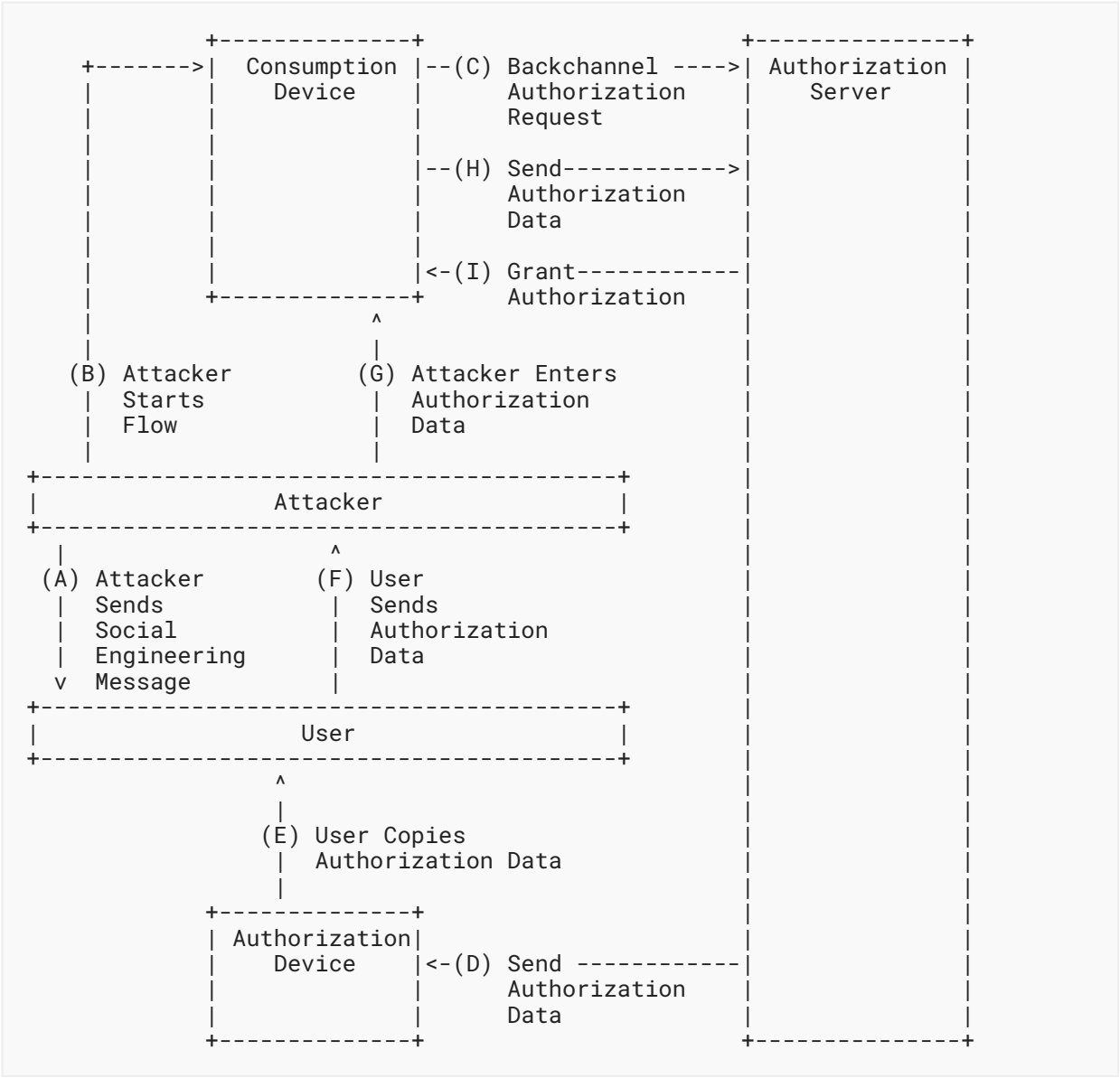

Figure 7: User-Transferred Authorization Data Pattern Exploit

(A) The attacker sends a social engineering message to prime the user for the authorization request they are about to receive, including instructions on what to do with the authorization data once they receive it.

(B) The attacker initiates the protocol on the Consumption Device (or by mimicking the Consumption Device) by starting a purchase, adding a device to a network, or accessing a service on the Consumption Device.

(C) The client on the Consumption Device requests user authorization on the backchannel from the Authorization Server.

(D) The Authorization Server sends authorization data (e.g., a 6-digit authorization code) to the Authorization Device. Examples of mechanisms that may be used to distribute the authorization data include text messages, email, or a mobile application. The authorization data may be presented as text or a QR code.

(E) The user is convinced by the social engineering message received from the attacker in step (A) and copies the authorization data received on the Authorization Device.

(F) The user forwards the authorization data to the attacker.

(G) The attacker enters the authorization data (e.g., a 6-digit authorization code) on the Consumption Device.

(H) The attacker's Consumption Device sends the authorization data to the Authorization Server.

(I) The Authorization Server grants authorization and issues access and refresh tokens to the Consumption Device, which is under the attacker's control. On completion of the exploit, the attacker gains access to the user's resources.

The unauthenticated channel may also be exploited in variations of the above scenario if there is no session maintained in the channel for steps C and G. In that case, a user (as opposed to the attacker) initiates the flow and is then convinced using social engineering techniques into sending the authorization data (e.g., a 6-digit authorization code) to the attacker instead of using it themselves. The authorization data may be represented as a QR code or text string (e.g., 6-digit authorization code). The attacker then starts the flow and uses the authorization data to obtain the privileges that would have been assigned to the user.

4.2. Cross-Device Session Transfer Exploits

Attackers exploit cross-device session transfer flows by using social engineering techniques typically used in phishing attacks to convince the user to authorize the transfer of a session and then send the session transfer code or QR code to the attacker. The absence of an authenticated channel between these two devices enables the attacker to use the session transfer code on their own device to obtain access to the session and access the user's data. These attacks are referred to as CDSP attacks.

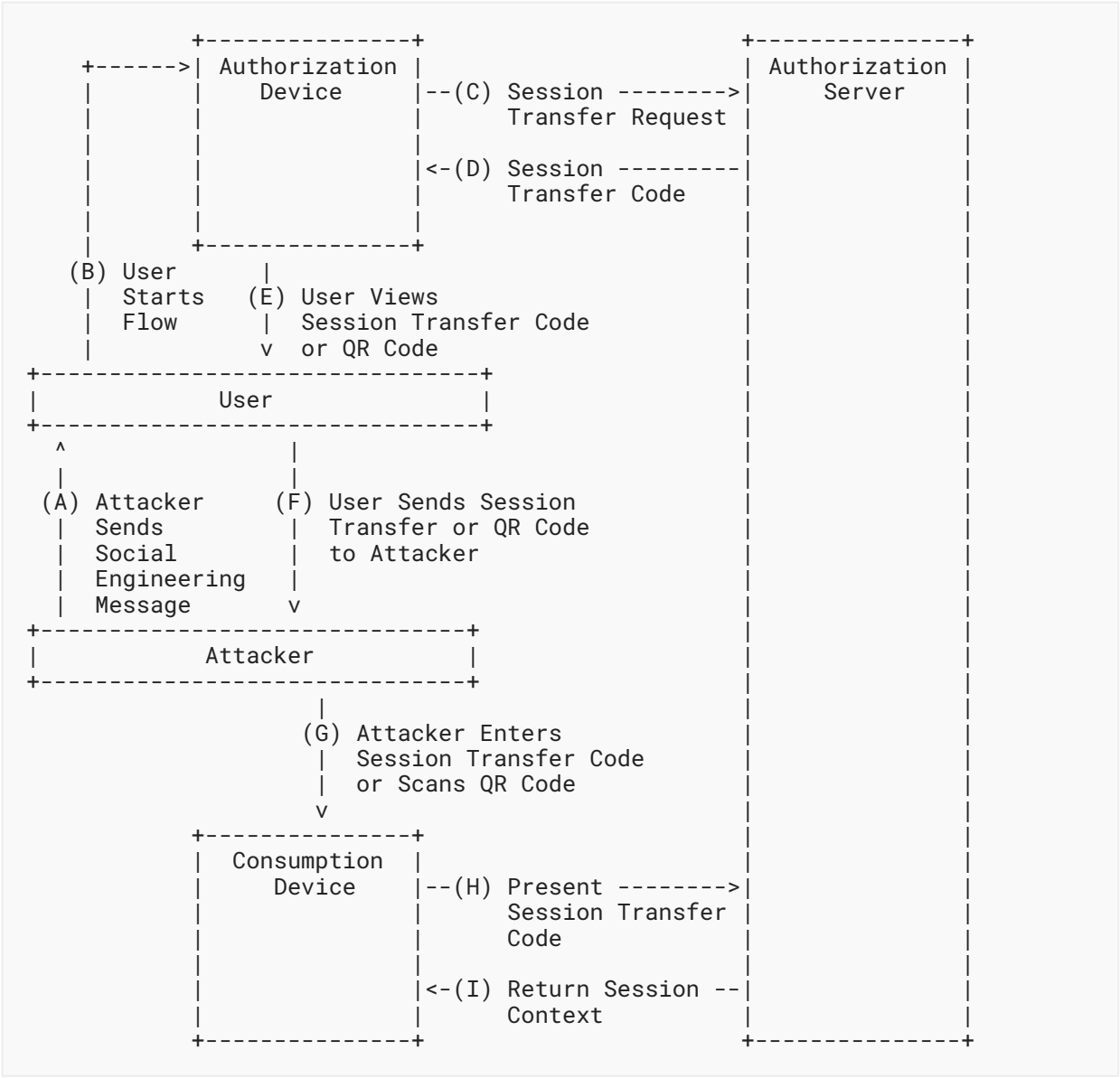

Figure 8: Cross-Device Session Transfer Pattern Exploit

(A) The attacker sends a social engineering message that convinces the user that they should authorize a session transfer including instructions on what to do with the QR code or session transfer code once they obtained it.

(B) The user starts the flow and is authenticated on their Authorization Device before they authorize the transfer of the session to the Consumption Device.

(C) The client on the Authorization Device requests a session transfer code from the Authorization Server.

(D) The Authorization Server responds with a session transfer code, which may be rendered as a QR code on the Authorization Device.

(E) The user views the session transfer code, which may be rendered as a QR code.

(F) The user sends the QR code or session transfer code to the attacker, following the instructions they received in step (A).

(G) Once the attacker receives the QR code, they scan it or enter it on their own Consumption Device.

(H) The client on the Consumption Device presents the session transfer code to the Authorization Server.

(I) The Authorization Server verifies the session transfer code and returns the session context information needed to resume the session on the Consumption Device. The attacker resumes the session on their own Consumption Device and is able to access the information that the user authorized on their Authorization Device in step (B).

4.3. Examples of Cross-Device Flow Exploits

The following examples illustrate these attacks in practical settings applied to the use cases described in Section 3.3. These examples show how the unauthenticated channel is exploited by attackers who can copy the QR codes and user codes, change the context in which they are presented using social engineering techniques, and mislead end users to grant consent to avail of services, access data, and make payments.

4.3.1. Example B1: Illicit Access to a Video Streaming Service (User-Transferred Session Data Pattern)

This exploit applies to the use case described in Section 3.3.1.

An attacker obtains a smart TV and attempts to access an online streaming service. The smart TV obtains a QR code from the streaming service Authorization Server and displays it on screen. The attacker copies the QR code and embeds it in an email that is sent to a large number of recipients. The email contains a message stating that the streaming service wants to thank them for their loyal support and by scanning the QR code, they will be able to add a bonus device to their account for no charge. One of the recipients opens the email and scans the QR code to claim the loyalty reward. The user performs multi-factor authentication, and when asked if they want a new device to be added to their account, they authorize the action. The attacker's device is now authorized to access the content and obtains an access and refresh token. The access token allows the attacker to access content, and the refresh token allows the attacker to obtain fresh tokens whenever the access token expires.

The attacker scales up the attack by emulating a new smart TV, obtaining multiple QR codes and widening the audience it sends the QR code to. Whenever a recipient scans the QR code and authorizes the addition of a new device, the attacker obtains an access and refresh token, which they sell for a profit.

4.3.2. Example B2: Illicit Access to Productivity Services (User-Transferred Session Data Pattern)

This exploit applies to the use case described in Section 3.3.2.

An attacker emulates an enterprise application (e.g., an interactive whiteboard) and initiates a cross-device flow by requesting a user code and URL from the Authorization Server. The attacker obtains a list of potential victims and sends an email informing users that their files will be deleted within 24 hours if they don't follow the link, enter the user code, and authenticate. The email reminds them that this is the third time that they have been notified and their last opportunity to prevent deletion of their work files. One or more employees respond by following the URL, entering the code, and performing multi-factor authentication. Throughout the authentication experience, the user is interacting with a trusted user experience, reinforcing the legitimacy of the request. Once these employees authorize access, the attacker obtains access and refresh tokens from the Authorization Server and uses them to access the users' files, perform lateral attacks to obtain access to other information, and continuously refresh the session by requesting new access tokens. These tokens may be exfiltrated and sold to third parties.

4.3.3. Example B3: Illicit Access to Physical Assets (User-Transferred Session Data Pattern)

This exploit applies to the use case described in Section 3.3.3.

An attacker copies a QR code from a bicycle locked in a bicycle rack in a city, prints it on a label, and places the label on a bicycle at the other end of the bicycle rack. A customer approaches the bicycle that contains the replicated QR code, scans the code, and authenticates before authorizing payment for renting the bicycle. The bicycle rack unlocks the bicycle containing the original QR code, and the attacker removes the bicycle before cycling down the street, while the customer is left frustrated that the bicycle they were trying to use is not being unlocked [NYC.Bike]. The customer proceeds to unlock another bicycle and lodges a complaint with the bicycle rental company.

4.3.4. Example B4: Illicit Transaction Authorization (Backchannel-Transferred Session Pattern)

These exploits apply to the use case described in Section 3.3.4.

4.3.4.1. Example B4.1: Bulk Authorization Request (Backchannel-Transferred Session Pattern)

An attacker obtains a list of user identifiers for a financial institution and triggers a transaction request for each of the users on the list. The financial institution's Authorization Server sends push notifications to each of the users, requesting authorization of a transaction. The vast majority of users ignore the request to authorize the transaction, but a small percentage grants authorization by approving the transaction.

4.3.4.2. Example B4.2: Fake Help Desk (Backchannel-Transferred Session Pattern)

An attacker obtains the contact information for a user and contacts them, pretending to be a representative of the user's financial institution. The attacker informs the user that there were a number of fraudulent transactions against their account and asks them to review these transactions by approving or rejecting them. The attacker then triggers a sequence of transactions. The user receives an authorization request for each transaction and declines them as they do not recognize them. The attacker then informs the user that they need to close the user's account and transfer all the funds to a new account to prevent further fraudulent transactions. The user receives another authorization request that they approve, or provides additional authorization information to the attacker, which enables the attacker to complete their attack and defraud the user.

4.3.5. Example B5: Illicit Network Join (Cross-Device Session Transfer Pattern)

This exploit applies to the use case described in Section 3.3.5.

An attacker creates a message to all employees of a company, claiming to be from a trusted technology provider investigating a suspected security breach. They ask employees to send them the QR code typically used to join a new device to the network, along with detailed steps on how to obtain the QR code. The employee, eager to assist, initiates the process to add a new mobile device to the network. They authenticate to the network and obtain a QR code. They send the QR code to the attacker. The attacker scans the QR code and adds their own device to the network. They use this device access as an entry point and perform lateral moves to obtain additional privileges and access to restricted resources.

4.3.6. Example B6: Illicit Onboarding (User-Transferred Session Data Pattern)

This exploit applies to the use case described in Section 3.3.6.

An attacker initiates an employee onboarding flow and obtains a QR code from the onboarding portal to invoke a digital wallet and present a verifiable credential attesting to a new employee's identity. The attacker obtains a list of potential new employees and sends an email informing them that it is time to present proof of their background check or government-issued ID. The new employee scans the QR code, invokes their digital wallet, and presents their credentials. Once the credentials are presented, the employee's account is activated. The employee portal accessed by the attacker to obtain the QR code displays a message to the attacker with instructions on how to access their account.

4.3.7. Example B7: Illicit Application Bootstrap (Cross-Device Session Transfer Pattern)

This exploit applies to the use case described in Section 3.3.7.

An attacker creates a message to all employees of a company, claiming to be from the company's IT service provider. They claim that they are trying to resolve an application performance issue and ask employees to send them the QR code typically used to transfer a session. The employee, eager to assist, initiates the process to transfer a session. They authenticate, obtain a QR code, and then send the QR code to the attacker. The attacker scans the QR code with their mobile phone and accesses the user's data and resources.

4.3.8. Example B8: Account Takeover (User-Transferred Authorization Data Pattern)

This exploit applies to the use case described in Section 3.3.8.

An attacker obtains a user's credentials for a Computer-Aided Design (CAD) application but cannot complete the authorization without the 6-digit authorization code sent to the user's mobile phone. The attacker triggers the authorization request, causing the Authorization Server to send a 6-digit authorization code to the user's mobile phone. The attacker then contacts the user, claiming to be from the CAD application's support desk and investigating a problem with the user's account, and asks the user to read back the code they have just received. Because the attacker has established the context for the code, the user provides it. The attacker enters the code in the CAD application and obtains access to the user's designs.

4.3.9. Example B9: Illicit Access to Administration Capabilities Through Consent Request Overload (Backchannel-Transferred Session Pattern)

This exploit applies to the use case described in Section 3.3.9.

An attacker attempts to access an administration portal repeatedly, generating a stream of authorization requests to the network administrator. The attempts are timed to occur while the administrator is asleep. The administrator is woken by the incoming requests on their phone, and in an attempt to stop the notifications, they accidentally approve access, and the attacker gains access to the portal.

4.3.10. Out of Scope

In all of the attack scenarios listed above, a user is misled or exploited. For other attacks, where the user is willingly colluding with the attacker, the threat model, security implications, and potential mitigations are very different. For example, a cooperating user can bypass software mitigations on their device, share access to hardware tokens with the attacker, and install additional devices to forward radio signals to circumvent proximity checks.

This document only considers scenarios where a user does not collude with an attacker.

5. Cross-Device Protocols and Standards

Cross-device flows that are subject to the attacks described earlier typically share the following characteristics:

1. The attacker can initiate the flow and manipulate the context of an authorization request. For example, the attacker can obtain a QR code or user code, or the attacker can request an authentication/authorization decision from the user.
2. The interaction between the Consumption Device and Authorization Device is unauthenticated. That is, it is left to the user to decide if the QR code, user code, or authentication request is being presented in a legitimate context.

Protocols that have been standardized (or are in the process of being standardized) that share these characteristics include:

- **IETF OAuth 2.0 device authorization grant [RFC8628]:** A standard to enable authorization on devices with constrained input capabilities (e.g., smart TVs, printers, and kiosks). In this protocol, the user code or QR code is displayed on the Consumption Device and entered on a second device (e.g., a mobile phone).
- **OpenID Foundation CIBA [CIBA]:** A standard developed in the OpenID Foundation that allows a device or service (e.g., a personal computer, smart TV, or kiosk) to request the OpenID Provider to initiate an authentication flow if it knows a valid identifier for the user. The user completes the authentication flow using a second device (e.g., a mobile phone). In this flow, the user does not scan a QR code or obtain a user code from the Consumption Device but is instead contacted by the OpenID Provider to complete the authentication using a push notification, email, text message, or any other suitable mechanism.
- **OpenID for Verifiable Credential Protocol Suite (Issuance, Presentation):** The OpenID for Verifiable Credentials enables cross-device scenarios by allowing users to scan QR codes to retrieve credentials (Issuance; see [OpenID.VCI]) or present credentials (Presentation; see [OpenID.VP]). The QR code is presented on a device that initiates the flow.
- **Self-Issued OpenID Provider v2 (SIOPv2):** A standard that allows end users to present self-attested or third-party attested attributes when used with OpenID for Verifiable Credential protocols. The user scans a QR code presented by the relying party to initiate the flow.

Cross-device protocols **SHOULD NOT** be used for same-device scenarios. If the Consumption Device and Authorization Device are the same device, protocols like OpenID Connect Core [OpenID.Core] and OAuth 2.0 authorization code grant as defined in [RFC6749] are more appropriate. If a protocol supports both same-device and cross-device modes (e.g., [OpenID.SIOPv2] and [OpenID.VP]), the cross-device mode **SHOULD NOT** be used for same-device scenarios. An Authorization Server **MAY** choose to block cross-device protocols used in same-device scenarios if it detects that the same device is used. Implementers should take into account that in environments that use Network Address Translation (NAT), multiple devices may appear to originate from the same network address, increasing the risk of incorrectly inferring that a cross-device flow is occurring on a single device.

The W3C Digital Credentials API [W3C.DCAPI] is a standard that defines a browser API for requesting and presenting verifiable credentials. The API is designed to be used in both same-device and cross-device scenarios. In cross-device scenarios, the API can be used to secure flows against the attacks presented in this document by ensuring proximity between the Consumption Device and Authorization Device (e.g., by leveraging Bluetooth Low Energy (BLE)). In same-device scenarios, the API can be used without proximity checks.

An Authorization Server **MAY** use device fingerprinting, network address, or other techniques to detect if a cross-device protocol is being used on the same device. If an implementer decides to use a cross-device protocol or a protocol with a cross-device mode in a same-device scenario, the mitigations recommended in this document **SHOULD** be implemented to reduce the risks that the unauthenticated channel is exploited.

6. Mitigating Against Cross-Device Flow Attacks

The unauthenticated channel between the Consumption Device and the Authorization Device allows attackers to change the context in which the authorization request is presented to the user. This shifts responsibility of authenticating the channel between the two devices to the end user. End users have "expertise elsewhere", are typically not security experts, and don't understand the protocols and systems they interact with. As a result, end users are poorly equipped to authenticate the channel between the two devices. Mitigations should focus on:

1. Minimizing reliance on the user to make decisions to authenticate the channel.
2. Providing better information with which to make decisions to authenticate the channel.
3. Recovering from incorrect channel authentication decisions by users.

To achieve the above outcomes, mitigating against CDCP attacks requires a three-pronged approach:

1. Reduce the risks of deployed protocols with practical mitigations.
2. Adopt or develop protocols that are less susceptible to these attacks where possible.
3. Provide analytical tools to assess vulnerabilities and effectiveness of mitigations.

6.1. Practical Mitigations

A number of protocols that enable cross-device flows that are susceptible to CDCP attacks are already deployed. The security profile of these protocols can be improved through practical mitigations that provide defense in depth that either:

1. Prevents the attack from being initiated.
2. Disrupts the attack once it is initiated.
3. Remediates or reduces the impact if the attack succeeds.

It is **RECOMMENDED** that one or more of the mitigations be applied when implementing a cross-device flow. Each mitigation, despite limitations in its effectiveness, provides an additional layer of security that may increase the difficulty of initiating an attack, disrupt attacks in progress, or reduce the impact of a successful attack.

6.1.1. Establish Proximity

The unauthenticated channel between the Consumption Device and Authorization Device allows attackers to obtain a QR code or user code in one location and display it in another location. Consequently, proximity-enforced cross-device flows are more resistant to CDCP attacks than proximity-less cross-device flows. Establishing proximity between the location of the Consumption Device and the Authorization Device limits an attacker's ability to launch attacks by sending user codes or QR codes to large numbers of users that are geographically distributed. Note that the Authorization Server typically cannot directly determine whether the Consumption Device and Authorization Device are physically close to each other. Instead, it must rely on the

surrounding systems, protocols in use, device capabilities, or information it obtains from other systems to establish or verify proximity. The Authorization Server can validate information it receives, but it cannot independently measure or enforce proximity on its own. There are a number of ways to establish proximity, each with its own implementation benefits and limitations:

- Physical connectivity: This is a good indicator of proximity but requires specific ports, cables, and hardware, and it may be challenging from a user experience perspective or may not be possible in certain settings (e.g., when USB ports are blocked or removed for security purposes). Physical connectivity may be better suited to dedicated hardware like FIDO devices that can be used with protocols that are resistant to the exploits described in this document. The use of physically connected devices may introduce additional security risks (e.g., data access or device compromise through malicious peripherals), the assessment and mitigation of which are beyond the scope of this document.
- Wireless proximity: Near Field Communications (NFC), Bluetooth Low Energy (BLE), and Ultra Wideband (UWB) services can be used to prove proximity between the two devices. NFC technology is widely deployed in mobile phones as part of payment solutions, but NFC readers are less widely deployed. BLE presents another alternative for establishing proximity but may present user experience challenges when setting up. UWB standards such as IEEE 802.15.4 and the IEEE 802.15.4z-2020 Amendment 1 enable secure ranging between devices and allow devices to establish proximity relative to each other [IEEE-802.15.4]. FIDO and WebAuthn-based cross-device flows leverage wireless proximity using BLE and are the **RECOMMENDED** approach for performing secure cross-device flows (see Section 6.2.3). For the presentation of digital credentials, the W3C Digital Credentials API [W3C.DCAPI] can be used.
- Shared network: Device proximity can be inferred by verifying that both devices are on the same network. This check may be performed by the Authorization Server by comparing the network addresses of the device where the code is displayed (Consumption Device) with that of the Authorization Device. Alternatively, the check can be performed on the device, provided that the network address is available. This could be achieved if the Authorization Server encodes the Consumption Device's network address in the QR code and uses a digital signature to prevent tampering with the code. This does require the wallet to be aware of the countermeasure and effectively enforce it. Note that it is common for a Consumption Device (e.g., a TV) to use a Wi-Fi connection while the Authorization Device (e.g., a phone) uses a mobile network. Though physically in proximity, they don't share a network, so other proximity checks are needed.
- Geolocation: Proximity can be established by comparing geolocation information derived from Global Navigation Satellite System (GNSS) coordinates or geolocation lookup of IP addresses and comparing proximity. Geolocation based on GNSS may vary in accuracy depending on the user's location and, when mapped to national or regional boundaries, may show a Consumption and Authorization Device in different locations if those devices are close to a border. Since relative position is more important than absolute location, implementations should consider relative location to both devices rather than absolute location when determining proximity. Geolocation based on IP addresses may be inaccurate along regional or national borders due to overlapping coverage by different network

providers from the respective regions. This may result in the Consumption Device being mapped to one region, while the Authorization Device may be on another network from another provider and mapped to another region. These inaccuracies may require restrictions to be at a more granular level (e.g., same city, country, region, or continent). Similar to the shared network checks, these checks may be performed by the Authorization Server or on the user's device, provided that the information encoded in a QR code is integrity protected using a digital signature.

Depending on the risk profile and the threat model in which a system is operating, it **MAY** be necessary to use more than one mechanism to establish proximity to raise the bar for any potential attackers. Proximity mechanisms that rely on establishing a user's location or identifying a user's device **SHOULD** be evaluated for their privacy implications within the context of a specific application or deployment.

> Note: There are scenarios that require that authorization takes place in a different location than the one in which the transaction is initiated. For example, there may be a primary and secondary credit card holder, and both can initiate transactions, but only the primary holder can authorize it. There is no guarantee that the primary and secondary holders are in the same location at the time of the authorization. In such cases, proximity can still serve as a risk signal. For example, while the primary and secondary holders may normally be located in the same city, a sudden presence in different continents may prompt the system to apply additional controls (e.g., transaction value limits or transaction velocity limits) or incorporate proximity information into a broader risk management decision.

Limitations: Proximity mechanisms make it harder to perform CDCP attacks. However, depending on how the proximity check is performed, an attacker may be able to circumvent the protection: The attacker can use a VPN to simulate a shared network or spoof a GNSS position. For example, the attacker can try to request the location of the end user's Authorization Device through browser APIs and then simulate the same location on their Consumption Device using standard debugging features available on many platforms. Relying on IP address mapping can degrade user experience when a VPN is used on the Consumption or Authorization Device. In such cases, the devices may appear to be in different locations, requiring additional user guidance or alternative mechanisms to establish proximity.

6.1.2. Short-Lived/Time-Bound QR or User Codes

The impact of an attack can be reduced by making QR or user codes short-lived. If an attacker obtains a short-lived code, the duration during which the unauthenticated channel can be exploited is reduced, potentially increasing the cost of a successful attack. This mitigation can be implemented on the Authorization Server without changes to other system components.

Limitations: There is a practical limit to how short a user code can be valid due to network latency and user experience limitations (time taken to enter a code, time to complete authentication, or time needed to re-enter codes or re-authenticate due to an error). More

sophisticated CDCP attacks counter the effectiveness of short-lived codes by convincing a user to respond to a phishing email and only request the QR or user code once the user clicks on the link in the phishing email [SQPHISH].

6.1.3. One-Time or Limited-Use Codes

By enforcing one-time use or limited use of user or QR codes, the Authorization Server can limit the impact of attacks where the same user code or QR code is sent to multiple victims. One-time use may be achieved by including a nonce or date stamp in the user code or QR code, which is validated by the Authorization Server when the user scans the QR code against a list of previously issued codes. This mitigation can be implemented on the Authorization Server without changes to other system components.

Limitations: Enforcing one-time use may be difficult in large globally distributed systems with low latency requirements, in which case short-lived tokens may be more practical. One-time use codes may also have an impact on the user experience. For example, a user may enter a code, but their session may be interrupted before the access request is completed. If the code is a one-time use code, they would need to restart the session and obtain a new code since they won't be allowed to enter the same code a second time. To avoid this, implementers **MAY** allow the same code to be presented a few times.

6.1.4. Unique Codes

By issuing unique user or QR codes, an Authorization Server can detect if the same codes are being repeatedly submitted. This may be interpreted as anomalous behavior, and the Authorization Server **MAY** choose to decline issuing access and refresh tokens if it detects the same codes being presented repeatedly. This may be achieved by maintaining a deny list that contains QR codes or user codes that were previously used. The Authorization Server **MAY** use a sliding window equal to the lifetime of a token if short-lived/time-bound tokens are used (see Section 6.1.2). This will limit the size of the deny list. This mitigation can be implemented on the Authorization Server without changes to other system components.

Limitations: Maintaining a deny list of previously redeemed codes, even for a sliding window, may have an impact on the latency of globally distributed systems. One alternative is to segment user codes by geography or region and maintain local deny lists.

6.1.5. Content Filtering

Attackers exploit the unauthenticated channel by changing the context of the user code or QR code and then sending a message to a user (email, text messaging, instant messaging, or other communication mechanisms). By deploying content filtering (e.g., anti-spam filter), these messages can be blocked and prevented from reaching the end users. It may be possible to fine-tune content-filtering solutions to detect artifacts like QR codes or user codes that are included in a message that is sent to multiple recipients in the expectation that at least one of the recipients will be convinced by the message and grant authorization to access restricted resources.

Limitations: Some scenarios may require legitimate retransmission of user data, QR codes, and authorization data (e.g., retries). To prevent the disruption of legitimate scenarios, content filters may use a threshold and allow a limited number of messages with the same QR or user codes to

be transmitted before interrupting the delivery of those messages. Content filtering may also be fragmented across multiple communications systems and communication channels (email, text messaging, instant messaging, or other communication mechanisms), making it harder to detect or interrupt attacks that are executed over multiple channels, unless there is a high degree of integration between content-filtering systems.

6.1.6. Detect and Remediate

The Authorization Server may be able to detect misuse of the codes due to repeated use as described in Section 6.1.4, as an input from a content-filtering engine as described in Section 6.1.5, or through other mechanisms such as reports from end users. If an Authorization Server determines that a user code or QR code is being used in an attack, it **MAY** choose to invalidate all tokens issued in response to these codes and make that information available through a token introspection endpoint (see [RFC7662]). In addition, it may notify resource servers to stop accepting these tokens or to terminate existing sessions associated with these tokens using Continuous Access Evaluation Protocol (CAEP) messages [CAEP] using the Shared Signals Framework (SSF) [SSF] or an equivalent notification system.

Limitations: Detection and remediation require that resource servers are integrated with security eventing systems or token introspection services. This may not always be practical for existing systems and may need to be targeted to the most critical resource services in an environment.

6.1.7. Trusted Devices

If an attacker is unable to initiate the protocol, they are unable to obtain a QR code or user code that can be leveraged for the attacks described in this document. By restricting the protocol to only be executed on devices trusted by the Authorization Server, it prevents attackers from using arbitrary devices or mimicking devices to initiate the protocol.

Authorization Servers **MAY** use different mechanisms to establish which devices they trust for cross-device flows. This includes limiting cross-device flows to specific device types such as interactive whiteboards or smart TVs, pre-registering devices with the Authorization Server, or only allowing cross-device flows on devices managed through device management systems. Device management systems may enforce policies that govern patching, version updates, on-device anti-malware deployment, revocation status, and device location amongst others. Trusted devices **MAY** have their identities rooted in hardware (e.g., a Trusted Platform Module (TPM) or equivalent technology).

By only allowing trusted devices to initiate cross-device flows, it requires the attacker to have access to such a device and maintain access in a way that does not result in the device's trust status from being revoked.

Mechanisms that identify a specific device **SHOULD** be evaluated for their privacy implications within the context of a specific application or deployment.

Limitations: An attacker may still be able to obtain access to a trusted device and use it to initiate authorization requests, making it necessary to apply additional controls and integrate with other threat detection and management systems that can detect suspicious behavior, such as repeated requests to initiate authorization or a high volume of service activation on the same device. An attacker may also spoof device identities or device types that are not cryptographically established or verified through attestation mechanisms.

6.1.8. Trusted Networks

An attacker can be prevented from initiating a cross-device flow protocol by only allowing the protocol to be initiated on a trusted network or within a security perimeter (e.g., a corporate network). A trusted network may be defined as a set of IP addresses, and joining the network is subject to security controls managed by the network operator, which may include only allowing trusted devices on the network, device management, user authentication, and physical access policies and systems. In some deployments, a trusted network may also be inferred using information supplied by a Subscriber Identity Module (SIM) or the network operator. By limiting protocol initiation to a specific network, the attacker needs to have access to a device on the network. This mitigation can be implemented on the Authorization Server without changes to other system components.

Limitations: Network-level controls may not always be feasible, especially when dealing with consumer scenarios where the network may not be under control of the service provider. Even if it is possible to deploy network-level controls, they **SHOULD** be used in conjunction with other controls outlined in this document to achieve defense in depth.

6.1.9. Limited Scopes

Authorization Servers **MAY** choose to limit the scopes they include in access tokens issued through cross-device flows where the unauthenticated channel between two devices is susceptible to being exploited. Including limited scopes lessens the impact in case of a successful attack. The decision about which scopes are included may be further refined based on whether the protocol is initiated on a trusted device or the user's location relative to the location of the Consumption Device. This mitigation can be implemented on the Authorization Server without changes to other system components.

Limitations: Limiting scopes reduces the impact of a compromise but does not avoid it. It **SHOULD** be used in conjunction with other mitigations described in this document.

6.1.10. Short-Lived Tokens

Another mitigation strategy includes limiting the life of the access and refresh tokens. The lifetime can be lengthened or shortened, depending on the user's location, the resources they are trying to access, or whether they are using a trusted device. Short-lived tokens do not prevent or disrupt the attack but serve as a remedial mechanism in case the attack succeeded. This mitigation can be implemented on the Authorization Server without changes to other system components.

Limitations: Short-lived tokens reduce the time window during which an attacker can benefit from a successful attack. This is most effective for access tokens. However, once an attacker obtains a refresh token, they can continue to request new access tokens as well as refresh tokens. Forcing the expiry of refresh tokens may cause the user to re-authorize an action more frequently, which results in a negative user experience.

6.1.11. Rate Limits

An attacker that engages in a scaled attack may need to request a large number of user codes (see the exploit described in Section 4.3.1) or initiate a large number of authorization requests (see the exploits described in Sections 4.3.4.1 and 4.3.9) in a short period of time. An Authorization Server **MAY** apply rate limits to minimize the number of requests it would accept from a client or send to a user in a limited time period.

Limitations: Rate limits are effective at slowing an attacker down and help to degrade scaled attacks, but they do not prevent more targeted attacks that are executed with lower volumes and velocity. Therefore, they should be used along with other techniques to provide defense in depth against cross-device attacks.

6.1.12. Sender-Constrained Tokens

Sender-constrained tokens limit the impact of a successful attack by preventing the tokens from being moved from the device on which the attack was successfully executed. This makes attacks where an attacker gathers a large number of access and refresh tokens on a single device and then sells them for profit more difficult, since the attacker would also have to export the cryptographic keys used to sender-constrain the tokens or be able to access them and generate signatures for future use. If the attack is being executed on a trusted device to a device with anti-malware, any attempts to exfiltrate tokens or keys may be detected and the device's trust status may be changed. Using hardware keys sender-constrained tokens will further reduce the ability of the attacker to move tokens to another device.

Limitations: Sender-constrained tokens, especially sender-constrained tokens that require proof of possession, raise the bar for executing the attack and profiting from exfiltrating tokens. Although a software proof-of-possession key is better than no proof-of-possession key, an attacker may still exfiltrate the software key. Hardware keys are harder to exfiltrate but come with additional implementation complexity. An attacker that controls the Consumption Device may still be able to exercise the key, even if it is in hardware. Consequently, the main protection derived from sender-constrained tokens is preventing tokens from being moved from the Consumption Device to another device, thereby making it harder to sell stolen tokens and profit from the attack.

6.1.13. User Education

Research shows that user education is effective in reducing the risk of phishing attacks [Baki2023]. The service provider **MAY** educate users on the risks of CDCP, as part of broader anti-phishing education, such as guidance to avoid clicking on links in emails or other unsolicited messages, as described by NIST in [NISTPhishing]. In addition, the service provider **MAY** provide out-of-band reinforcement on the context and conditions under which an authorization grant

may be requested. For example, if the service provider does not send emails containing QR codes that request users to grant authorization, this expectation may be reinforced through marketing communications and anti-fraud awareness campaigns. The service provider **MAY** also reinforce these user education messages through in-app experiences. In [PCRSM2023], it is proposed that users be advised to verify the trustworthiness of the source of a QR code, for example, by confirming that the connection is protected using TLS or that the URL belongs to the Authorization Server.

Limitations: Although user education helps to raise awareness and reduce the overall risk to users, it is insufficient on its own to mitigate CDCP attacks. In particular, carefully designed phishing attacks can be practically indistinguishable from benign authorization flows even for well-trained users. User education **SHOULD** therefore be used in conjunction with other controls described in this document.

6.1.14. User Experience

The user experience **SHOULD** preserve the context within which the protocols were initiated and communicate this clearly to the user when they are asked to authorize, authenticate, or present a credential. In preserving the context, it should be clear to the user who invoked the flow, why it was invoked, and what the consequence of completing the authorization, authentication, or credential presentation is. The user experience **SHOULD** reinforce the message that unless the user initiated the authorization request, or was expecting it, they should decline the request.

This information **MAY** be communicated graphically or in a simple message (e.g., "It looks like you are trying to access your files on a digital whiteboard in your city center office. Click here to grant access to your files. If you are not trying to access your files, you should decline this request and notify the security department").

The user interface **SHOULD** provide an obvious and unambiguous way for the user to decline or cancel a request. To avoid accidental authorization grants, the "decline" option **SHOULD** be the default option or given similar prominence in the user experience as the "grant" option.

If the user uses an application on a mobile device to scan a QR code, the application **MAY** display information advising the user under which conditions they should expect to be asked to scan a QR code and under which circumstances they should never scan a QR code (e.g., display a message that the QR code will only be displayed on kiosks within trusted locations or on trusted websites hosted on a specific domain, and never in email or other media and locations).

The user experience **MAY** include information to further educate the user on CDCP attacks and reinforce the conditions under which authorization grants may be requested.

Limitations: Improvements to user experience on their own are unlikely to be sufficient and **SHOULD** be used in conjunction with other controls described in this document.

6.1.15. Authenticate then Initiate

By requiring a user to authenticate on the Consumption Device with a phishing-resistant authentication method before initiating a cross-device flow, the server can prevent an attacker from initiating a cross-device flow and obtaining QR codes or user codes. For example, a banking

application may initiate a cross-device authorization request using the Backchannel-Transferred Session Pattern only after the user is already authenticated and attempts a high-value transaction. This prevents the attacker from initiating a cross-device authorization request and obtaining a QR code or a user code that they can use to mislead an unsuspecting user. This requires that the Consumption Device has sufficient input capabilities to support a phishing-resistant authentication mechanism, which may in itself negate the need for a cross-device flow.

Limitations: This mitigation is limited to Consumption Devices capable of supporting phishing-resistant authentication mechanisms. Authenticating on the Consumption Device before starting a cross-device flow does not prevent the attacks described in Sections 4.3.5 and 4.3.7, and it is **RECOMMENDED** that additional mitigations described in this document be used if the cross-device flows are used in scenarios such as those described in Sections 3.3.5 and 3.3.7.

6.1.16. Request Initiation Verification

The user **MAY** be asked to confirm if they initiated an authentication or authorization request by sending a one-time password (OTP) or PIN to the user's Authorization Device and asking them to enter it on the Consumption Device to confirm the request. If the request was initiated without the user's consent, they would receive an OTP or PIN out of context, which may raise suspicion for the user. In addition, they would not have information on where to enter the OTP or PIN. The user experience on the Authorization Device **MAY** reinforce the risk of receiving an out-of-context OTP or PIN and provide information to the user on how to report an unauthorized authentication or authorization request.

Limitations: The additional verification step may reduce the overall usability of the system as it is one more thing users need to do right. Attackers may combine conventional phishing attacks and target users who respond to those messages with an interactive attack that sets the expectation with the user that they will have to provide the OTP or PIN, in addition to granting authorization for the request.

6.1.17. Request Binding with Out-of-Band Data

In the User-Transferred Session Data Pattern, users **MAY** enter out-of-band information on the Consumption Device to start the authorization process. The out-of-band data entered by the user **MAY** then be included in the QR code, which is displayed on the Consumption Device. When the QR code is scanned by the Authorization Device, the out-of-band data is verified by the user or by the Authorization Device. The out-of-band data could be any attribute that the user or Authorization Device can retrieve during the authorization process. Examples include a serial number, OTP or PIN, location, or any other data that the user or the Authorization Device can recall or retrieve during the authorization process (see [MPRCS2020] and [PCRSM2023]).

Limitations: A sophisticated attacker may include an additional step in their attack where they create a phishing attack that gathers the out-of-band data from the user before initiating the authorization request. The additional step could also have a negative impact on the usability level of the solution.

6.1.18. Practical Mitigation Summary

The practical mitigations described in this section can, within the limitations described, prevent or substantially increase the difficulty of initiating attacks, disrupt attacks once they start, or reduce the impact of or remediate an attack if it succeeds. When one or more of these mitigations are combined, the overall security profile of a cross-device flow improves significantly. The following table provides a summary of these mitigations:

Mitigation	Prevent	Disrupt	Recover
Establish Proximity	X	X	
Short-Lived/Time-Bound Codes		X	
One-Time or Limited-Use Codes		X	
Unique Codes		X	
Content Filtering		X	
Detect and Remediate			X
Trusted Devices	X		
Trusted Networks	X		
Limited Scopes			X
Short-Lived Tokens			X
Rate Limits	X	X	
Sender-Constrained Tokens			X
User Education	X		
User Experience	X		
Authenticate then Initiate	X		
Request Initiation Verification		X	
Request Binding with Out-of-Band Data		X	

Table 1: Practical Mitigation Summary

6.2. Protocol Selection

Some cross-device protocols are more susceptible to the exploits described in this document than others. In this section, we will compare three different cross-device protocols in terms of their susceptibility to exploits focused on the unauthenticated channel, the prerequisites to implement and deploy them, along with guidance on when it is appropriate to use them.

6.2.1. IETF OAuth 2.0 Device Authorization Grant

6.2.1.1. Description

[RFC8628] is a standard to enable authorization on devices with constrained input capabilities (e.g., smart TVs, printers, and kiosks). In this protocol, the user code or QR code is displayed or made available on the Consumption Device (smart TV) and entered on a second device (e.g., a mobile phone).

6.2.1.2. Susceptibility

There are several reports in the public domain outlining how the unauthenticated channel may be exploited to execute a CDCP attack (see [ARTDCPHISH], [DCFLOWPHISH], [NEWDCPHISH], [DEFCON29], [DCATTACK], and [SQPHISH]).

6.2.1.3. Device Capabilities

There are no assumptions in the protocol about underlying capabilities of the device, making it a "least common denominator" protocol that is expected to work on the broadest set of devices and environments.

6.2.1.4. Mitigations

In addition to the Security Considerations section in [RFC8628], it is **RECOMMENDED** that one or more of the mitigations outlined in this document be considered, especially mitigations that can help establish proximity or prevent attackers from obtaining QR or user codes.

6.2.1.5. When to Use

Only use this protocol if other cross-device protocols are not viable due to device or system constraints. Avoid using if the protected resources are sensitive, high value, or business critical. Always deploy additional mitigations like proximity or only allow with pre-registered devices. Do not use for same-device scenarios (e.g., if the Consumption Device and Authorization Device are the same device).

6.2.2. OpenID Foundation Client-Initiated Backchannel Authentication (CIBA)

6.2.2.1. Description

CIBA [CIBA] is a standard developed in the OpenID Foundation that allows a device or service (e.g., a personal computer, smart TV, or kiosk) to request the OpenID Provider to initiate an authentication flow if it knows a valid identifier for the user. The user completes the authentication flow using a second device (e.g., a mobile phone). In this flow, the user does not

scan a QR code or obtain a user code from the Consumption Device, but is instead contacted by the OpenID Provider to complete the authentication using a push notification, email, text message, or any other suitable mechanism.

6.2.2.2. Susceptibility

CIBA is less susceptible to unauthenticated channel attacks, but it is still vulnerable to attackers who know or can guess the user identifier and initiate an attack as described in Section 4.3.4.1.

6.2.2.3. Device Capabilities

There is no requirement on the Consumption Device to support specific hardware. The Authorization Device must be registered/associated with the user, and it must be possible for the Authorization Server to trigger an authorization on this device.

6.2.2.4. Mitigations

In addition to the Security Considerations section in [CIBA], it is **RECOMMENDED** that one or more of the mitigations outlined in this document be considered, especially mitigations that can help establish proximity or prevent attackers from initiating authorization requests.

6.2.2.5. When to Use

Use CIBA instead of the device authorization grant if it is possible for the Consumption Device to obtain a user identifier on the Consumption Device (e.g., through an input or selection mechanism) and if the Authorization Server can trigger an authorization on the Authorization Device. Do not use for same-device scenarios (e.g., if the Consumption Device and Authorization Device are the same device).

6.2.3. FIDO2/WebAuthn

6.2.3.1. Description

FIDO2/WebAuthn is a stack of standards developed in the FIDO Alliance and W3C, respectively, which allow for origin-bound, phishing-resistant user authentication using asymmetric cryptography that can be invoked from a web browser or native client. Version 2.2 of the FIDO Client to Authenticator Protocol (CTAP) supports a new cross-device authentication protocol, called "hybrid transports", which enables an external device, such as a phone or tablet, to be used as a roaming authenticator for signing in to the primary device, such as a personal computer. This is commonly called FIDO Cross-Device Authentication (CDA). CTAP 2.2 hybrid transports is implemented by the client and authenticator platforms.

When a user wants to authenticate using their mobile device (authenticator) for the first time, they need to link their authenticator to their main device. This is done using a scan of a QR code. When the authenticator scans the QR code, the device sends an encrypted BLE advertisement containing keying material and a tunnel ID. The main device (CTAP client) and authenticator both establish connections to the web service, and the normal CTAP protocol exchange occurs.

If the user chooses to keep their authenticator linked with the main device, the QR code link step is not necessary for subsequent use. The user will receive a push notification on the authenticator.

6.2.3.2. Susceptibility

The CDA flow proves proximity by leveraging BLE advertisements for service establishment, significantly reducing the susceptibility to any of the exploits described in examples B1-B6 in Section 4.3.

6.2.3.3. Device Capabilities

Both the Consumption Device and the authenticator require BLE support and access to the internet. The Consumption Device must support both the WebAuthn API [W3CWebAuthn] (or a platform-specific WebAuthn abstraction for native apps) and the FIDO CTAP, specifically version 2.2 with hybrid transports [FIDOCTAP22]. The device serving as the FIDO authenticator must also support CTAP 2.2 or later to be used as a cross-device authenticator.

6.2.3.4. Mitigations

FIDO CDA establishes proximity through the use of BLE, reducing the need for additional mitigations. An implementer **MAY** still choose to implement additional mitigations as described in this document.

6.2.3.5. When to Use

FIDO2/WebAuthn **SHOULD** be used for cross-device authentication scenarios whenever the devices are capable of doing so and a suitable FIDO credential is not available on the Consumption Device. It **MAY** be used as an authentication method with the authorization code grant [RFC6749] and Proof Key for Code Exchange (PKCE) [RFC7636], to grant authorization to a Consumption Device (e.g., a smart TV or interactive whiteboard) using a device serving as the FIDO authenticator (e.g., a mobile phone) for authentication. This combination of FIDO2/WebAuthn and authorization code flow with PKCE enables cross-device authorization flows, without the risks posed by the device authorization grant [RFC8628].

6.2.4. Protocol Selection Summary

The FIDO CDA flow provides the best protection against attacks on the unauthenticated channel for cross-device flows. It can be combined with OAuth 2.0 and OpenID Connect protocols for standards-based authorization and authentication flows. If FIDO2/WebAuthn support is not available, CIBA provides an alternative, provided that there is a channel through which the Authorization Server can contact the end user. Examples of such a channel include device push notifications, email, or text messages that the user can access from their device. If CIBA is used, additional mitigations to enforce proximity and initiate transactions from trusted devices or trusted networks **SHOULD** be considered. The OAuth 2.0 device authorization grant provides the most flexibility and has the lowest requirements on devices used, but it is **RECOMMENDED** that it only be used when additional mitigations are deployed to prevent attacks that exploit the unauthenticated channel between devices.

6.3. Foundational Pillars

Experience with web authorization and authentication protocols such as OAuth and OpenID Connect has shown that securing these protocols can be hard. The major reason for this is that the landscape in which they are operating -- the web infrastructure with browsers, servers, and the underlying network -- is complex, diverse, and ever-evolving.

As is the case with other kinds of protocols, it can be easy to overlook vulnerabilities in this environment. One way to reduce the chances of hidden security problems is to use mathematical-logical models to describe the protocols, their environments, and their security goals, and then use these models to try to prove security. This approach is what is usually subsumed as "formal security analysis".

There are two major strengths of formal analysis. First, finding new vulnerabilities does not require creativity (i.e., new classes of attacks can be uncovered even if no one thought of these attacks before). In a faithful model, vulnerabilities become clear during the proof process or even earlier. Second, formal analysis can exclude the existence of any attacks within the boundaries of the model (e.g., the protocol layers modeled, the level of detail and functionalities covered, the assumed attacker capabilities, and the formalized security goals).

As a downside, there is usually a gap between the model (which necessarily abstracts away from details) and implementations. In other words, implementations can introduce flaws where the model does not have any. Nonetheless, for protocol standards, formal analysis can help to ensure that the specification is secure when implemented correctly.

There are various different approaches to formal security analysis, and each brings its own strengths and weaknesses. For example, models differ in the level of detail in which they can capture a protocol (granularity and expressiveness), in the kind of statements they can produce, and whether the proofs can be assisted by tools or have to be performed manually.

The following works have been identified as relevant to the analysis of cross-device flows:

- In "Formal analysis of self-issued OpenID providers" [Bauer2022], the protocol of [OpenID.SIOPv2] was analyzed using the Web Infrastructure Model (WIM). The WIM is specifically designed for the analysis of web authentication and authorization protocols. While it is a manual (pen-and-paper) model, it captures details of browsers and web interactions to a degree that is hard to match in automated models. In previous works, previously unknown flaws in OAuth, OpenID Connect, and OpenID FAPI were discovered using the WIM. In the analysis of a cross-device SIOPv2 flow in [Bauer2022], the request replay attack already described in Section 13.3 of [OpenID.SIOPv2] was confirmed in the model. A mitigation was implemented based on a so-called Cross-Device Stub, essentially a component that serves to link the two devices before the protocol flow starts. This can be seen as an implementation of a trusted device relationship as described in Section 6.1.7. The mitigation was shown to be effective in the model.

- In "Security analysis of the Grant Negotiation and Authorization Protocol" [Helmschmidt2022], an analysis of a draft of the Grant Negotiation and Authorization Protocol (GNAP) [RFC9635] was performed using the Web Infrastructure Model (WIM). The same attack as in [Bauer2022] was found to apply to GNAP as well. In this case, a model of a "careful user" (see Section 6.1.13) was used to show that the attack can be prevented (at least in theory) by the user.
- In "The Good, the Bad and the (Not So) Ugly of Out-of-Band Authentication with eID Cards and Push Notifications: Design, Formal and Risk Analysis" [MPRCS2020], Pernpruner et al. formally analyzed an authentication protocol relying on push notifications delivered to an out-of-band device to approve the authentication attempt on the primary device (the Backchannel-Transferred Session Pattern in Section 3.1.2). The analysis was performed using the specification language ASLan++ and the model checker SATMC. According to the results of the analysis, they identified and defined the category of "implicit attacks", which manage to deceive users into approving a malicious authentication attempt through social engineering techniques, thus not compromising all the authentication factors involved; these attacks are aligned with the definition of CDCP attacks.
- In "An Automated Multi-Layered Methodology to Assist the Secure and Risk-Aware Design of Multi-Factor Authentication Protocols" [PCRSM2023], Pernpruner et al. defined a multi-layered methodology to analyze multi-factor authentication protocols at different levels of granularity. They leveraged their methodology to formally analyze a protocol relying on a QR code that has to be scanned on a secondary device to approve the authentication attempt on the primary device (the User-Transferred Session Data Pattern in Section 3.1.1). Given the results of the analysis, they proposed some practical mitigations to either prevent or reduce the risk of successful attacks, such as those described in Sections 6.1.13, 6.1.16, and 6.1.17.

7. Security Considerations

Security considerations are described in Sections 2 and 6.

8. IANA Considerations

This document has no IANA actions.

9. Conclusion

Cross-device flows enable authorization on devices with limited input capabilities, allow for secure authentication when using public or shared devices, provide a path toward multi-factor authentication, and provide the convenience of a single, portable credential store.

The popularity of cross-device flows attracted the attention of attackers that exploit the unauthenticated channel between the Consumption Device and Authorization Device using techniques commonly used in phishing attacks. These CDCP attacks allow attackers to obtain access and refresh tokens, rather than authentication credentials, resulting in access to resources even if the user used multi-factor authentication.

To address these attacks, we propose a three-pronged approach that includes deploying practical mitigations to safeguard protocols that are already deployed, providing guidance on when to use different protocols (including protocols that are not susceptible to these attacks), and introducing formal methods to evaluate the impact of mitigations and find additional issues.

10. References

10.1. Normative References

[CAEP] Tulshibagwale, A. and T. Cappalli, "OpenID Continuous Access Evaluation Profile 1.0", August 2025, <https://openid.net/specs/openid-caep-1_0-final.html>.

[CIBA] Rodriguez, G. F., Walter, F., Nennker, A., Tonge, D., and B. Campbell, "OpenID Connect Client-Initiated Backchannel Authentication Flow - Core 1.0", September 2021, <https://openid.net/specs/openid-client-initiated-backchannel-authentication-core-1_0.html>.

[FIDOCTAP22] Bradley, J., Jones, M.B., Kumar, A., Lindemann, R., Verrept, S., and D. Waite, "Client to Authenticator Protocol (CTAP)", July 2025, <https://fidoalliance.org/specs/fido-v2.2-ps-20250714/fido-client-to-authenticator-protocol-v2.2-ps-20250714.html>.

[IEEE-802.15.4] IEEE, "IEEE Standard for Low-Rate Wireless Networks", IEEE Std 802.15.4-2024, DOI 10.1109/IEEESTD.2024.10794632, 2024, <https://doi.org/10.1109/IEEESTD.2024.10794632>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/info/rfc6749>.

[RFC7636] Sakimura, N., Ed., Bradley, J., and N. Agarwal, "Proof Key for Code Exchange by OAuth Public Clients", RFC 7636, DOI 10.17487/RFC7636, September 2015, <https://www.rfc-editor.org/info/rfc7636>.

[RFC7662] Richer, J., Ed., "OAuth 2.0 Token Introspection", RFC 7662, DOI 10.17487/RFC7662, October 2015, <https://www.rfc-editor.org/info/rfc7662>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8628] Denniss, W., Bradley, J., Jones, M., and H. Tschofenig, "OAuth 2.0 Device Authorization Grant", RFC 8628, DOI 10.17487/RFC8628, August 2019, <https://www.rfc-editor.org/info/rfc8628>.

[SSF] Tulshibagwale, A., Cappalli, T., Scurtescu, M., Backman, A., Bradley, J., and S. Miel, "OpenID Shared Signals Framework Specification 1.0", August 2025, <https://openid.net/specs/openid-sharedsignals-framework-1_0-final.html>.

[W3CWebAuthn] Cappalli, T., Ed., Kumar, A., Ed., Lundberg, E., Ed., Miller, M., Ed., Pascoe, Ed., and N. Satragno, "Web Authentication: An API for accessing Public Key Credentials Level 3", W3C Candidate Recommendation Snapshot, January 2025, <https://www.w3.org/TR/2026/CR-webauthn-3-20260526/>. Latest version available at <https://www.w3.org/TR/webauthn-3/>.

10.2. Informative References

[ARTDCPHISH] Cooke, B., "The Art of the Device Code Phish", July 2021, <https://0xboku.com/2021/07/12/ArtOfDeviceCodePhish.html>.

[Baki2023] Baki, S. and R. M. Verma, "Sixteen Years of Phishing User Studies: What Have We Learned?", IEEE Transactions on Dependable and Secure Computing, vol. 20, no. 2, pp. 1200-1212, 2023, <https://doi.org/10.1109/TDSC.2022.3151103>.

[Bauer2022] Bauer, C., "Formal analysis of self-issued OpenID providers", Master's Thesis, University of Stuttgart, 2022, <https://elib.uni-stuttgart.de/handle/11682/12417>.

[DCATTACK] Secureworks Counter Threat Unit Research Team, "OAuth's Device Code Flow Abused in Phishing Attacks", June 2021, <https://www.sophos.com/en-us/blog/oauths-device-code-flow-abused-in-phishing-attacks>.

[DCFLOWPHISH] Min, D., "Microsoft 365 OAuth Device Code Flow and Phishing", August 2021, <https://www.optiv.com/insights/source-zero/blog/microsoft-365-oauth-device-code-flow-and-phishing>.

[DEFCON29] Hwong, J., "New Phishing Attacks Exploiting OAuth Authentication Flows (DEFCON 29)", DEF CON 29, Video, August 2021, <https://www.youtube.com/watch?v=9slRYvpKHp4>.

[Helmschmidt2022] Helmschmidt, F., "Security analysis of the Grant Negotiation and Authorization Protocol", Master's Thesis, University of Stuttgart, 2022, <https://elib.uni-stuttgart.de/handle/11682/12220>.

[MPRCS2020] Pernpruner, M., Carbone, R., Ranise, S., and G. Sciarretta, "The Good, the Bad and the (Not So) Ugly of Out-of-Band Authentication with eID Cards and Push Notifications: Design, Formal and Risk Analysis", Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy, pp. 223-234, DOI 10.1145/3374664.3375727, 2020, <https://doi.org/10.1145/3374664.3375727>.

[NEWDCPHISH] Syynimaa, N., "Introducing a new phishing technique for compromising Office 365 accounts", October 2020, <https://aadinternals.com/post/phishing/>.

[NISTGlossary] NIST, "NIST Computer Security Resource Center Glossary", <https://csrc.nist.gov/glossary>.

[NISTPhishing] NIST, "NIST Small Business Cybersecurity Fact Sheet: Phishing", <https://www.nist.gov/system/files/documents/2024/03/12/Phishing_SMB%20FactSheet_2024_Final.pdf>.

[NYC.Bike] Byrne, K. J., "Citi Bikes being swiped by joyriding scammers who have cracked the QR code", New York Post, August 2021, <https://nypost.com/2021/08/07/citi-bikes-being-swiped-by-joyriding-scammers-who-have-cracked-the-qr-code/>.

[OpenID.Core] Sakimura, N., Bradley, J., Jones, M. B., de Medeiros, B., and C. Mortimore, "OpenID Connect Core 1.0 incorporating errata set 2", December 2023, <https://openid.net/specs/openid-connect-core-1_0.html>.

[OpenID.SIOPv2] Yasuda, K., Jones, M., and T. Lodderstedt, "Self-Issued OpenID Provider v2 - draft 13", November 2023, <https://openid.net/specs/openid-connect-self-issued-v2-1_0.html>.

[OpenID.VCI] Lodderstedt, T., Yasuda, K., Looker, T., and P. Bastian, "OpenID for Verifiable Credential Issuance 1.0", September 2025, <https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0.html>.

[OpenID.VP] Terbu, O., Lodderstedt, T., Yasuda, K., Fett, D., and J. Heenan, "OpenID for Verifiable Presentations 1.0", July 2025, <https://openid.net/specs/openid-4-verifiable-presentations-1_0.html>.

[PCRSM2023] Pernpruner, M., Carbone, R., Sciarretta, G., and S. Ranise, "An Automated Multi-Layered Methodology to Assist the Secure and Risk-Aware Design of Multi-Factor Authentication Protocols", IEEE Transactions on Dependable and Secure Computing, vol. 21, no. 4, pp. 1935-1950, 2023, <https://doi.org/10.1109/TDSC.2023.3296210>.

[RFC9635] Richer, J., Ed. and F. Imbault, "Grant Negotiation and Authorization Protocol (GNAP)", RFC 9635, DOI 10.17487/RFC9635, October 2024, <https://www.rfc-editor.org/info/rfc9635>.

[SQPHISH] Talebzadeh, K. and N. Romsdahl, "SquarePhish: Advanced phishing tool combines QR codes and OAuth 2.0 device code flow", Help Net Security, Video, August 2022, <https://www.helpnetsecurity.com/2022/08/11/squarephish-video/>.

[W3C.DCAPI] Caceres, M., Ed., Cappalli, T., Ed., and M. Yosef, Ed., "Digital Credentials API", W3C Editor's Draft, 16 July 2026, <https://w3c-fedid.github.io/digital-credentials/>. Latest version available at <https://www.w3.org/TR/digital-credentials/>.

Contributors

The authors would like to thank Tim Cappalli, Nick Ludwig, Adrian Frei, Nikhil Reddy Boreddy, Bjorn Hjelm, Joseph Heenan, Brian Campbell, Damien Bowden, Kristina Yasuda, Tim Würtele, Karsten Meyer zu Selhausen, Maryam Mehrnezhad, Marco Pernpruner, Giada Sciarretta, Dean

H. Saxe, Roy Williams, Aaron Parecki, George Fletcher, Hannes Tschofenig, Dan Moore, Deb Cooley, Paul Kyzivat, David Mandelberg, Jim Fenton, Bing Liu, Mohamed Boucadair, Mike Bishop, Roman Danyliw, and others for their valuable input, feedback, and general support of this work.

Authors' Addresses

Pieter Kasselman
Defakto Security
Email: prkasselman@gmail.com

Daniel Fett
Authlete
Email: mail@danielfett.de

Filip Skokan
Okta
Email: panva.ip@gmail.com