Workgroup: Web Authorization Protocol
Internet-Draft:
draft-ietf-oauth-cross-device-security-01
Published: 13 March 2023
Intended Status: Best Current Practice
Expires: 14 September 2023
Authors: P. Kasselman D. Fett F. Skokan
Microsoft yes.com Okta

Cross-Device Flows: Security Best Current Practice

Abstract

This document describes threats against cross-device flows along with near term mitigations, protocol selection guidance and the analytical tools needed to evaluate the effectiveness of these mitigations. It serves as a security guide to system designers, architects, product managers, security specialists, fraud analysts and engineers implementing cross-device flows.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the Web Authorization Protocol Working Group mailing list (oauth@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/oauth/.

Source for this draft and an issue tracker can be found at https://github.com/oauth-wg/oauth-cross-device-security.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 14 September 2023.

Copyright Notice

Table of Contents

1. Introduction

Cross-device flows enable a user to initiate an authorization flow on one device (the initiating device) and then use a second, personally trusted, device (authorization device) to authorize access to a resource (e.g., access to a service).

These flows are increasingly popular and typically involve using a mobile phone to scan a QR code or enter a user code displayed on an initiating device (e.g., Smart TV, Kiosk, Personal Computer etc).

The channel between the initiating device and the authorization device is unauthenticated and relies on the user's judgment to decide whether to trust a QR code, user code, or the authorization request pushed to their authorization device.

Several publications have emerged in the public domain ([Exploit1], [Exploit2], [Exploit3], [Exploit4], [Exploit5], [Exploit6]), describing how the unauthenticated channel can be exploited using social engineering techniques borrowed from phishing. Unlike traditional 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.

In order to defend against these attacks, this document outlines three responses:

1. For protocols that are susceptible to unauthenticated channel exploits, deploy practical mitigations.

2. Select protocols that are not susceptible to unauthenticated channel exploits when possible.

3. Conduct formal analysis of cross-device flows to assess susceptibility to these attacks and the effectiveness of the proposed mitigations.

1.1. 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 endpoint", "authorization request", "authorization response", "token endpoint", "grant type", "access token request", "access token response", and "client" defined by The OAuth 2.0 Authorization Framework [RFC6749].

2. Cross Device Flow Concepts

In a cross-device flow, a user starts a scenario on the initiating device (e.g., a smart TV) and then uses an authorization device (e.g., a smartphone) to authorize access to a resource (e.g., access to a streaming service).

Cross device 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, etc).

* 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 Initiating Device to the Authorization Device.

* User transferred: In the first variant, the user initiates the authorization process with the authorization server by copying information from the initiating device to the authorization device, before authorizing an action. For example the user may read a code displayed on the initiating device and enter it on the authorization device, or they may scan a QR code displayed in the initiating device with the authorization device.

* Client transferred: In the second variant, the OAuth client on the initiating device is responsible for initiating authorization on the authorization device via a backchannel with the authroization server.

* Hybrid: In the third variant, the OAuth client on the initiating device triggers the authorization request via a backchannel with the Authorization Server. An access code is displayed on the Authorization device, which the user enters on the initiating device.

2.1. User Transferred Pattern

An example of a cross-device flow that relies on the user copying information from the Initiating Device to the Authorization Device is shown below:

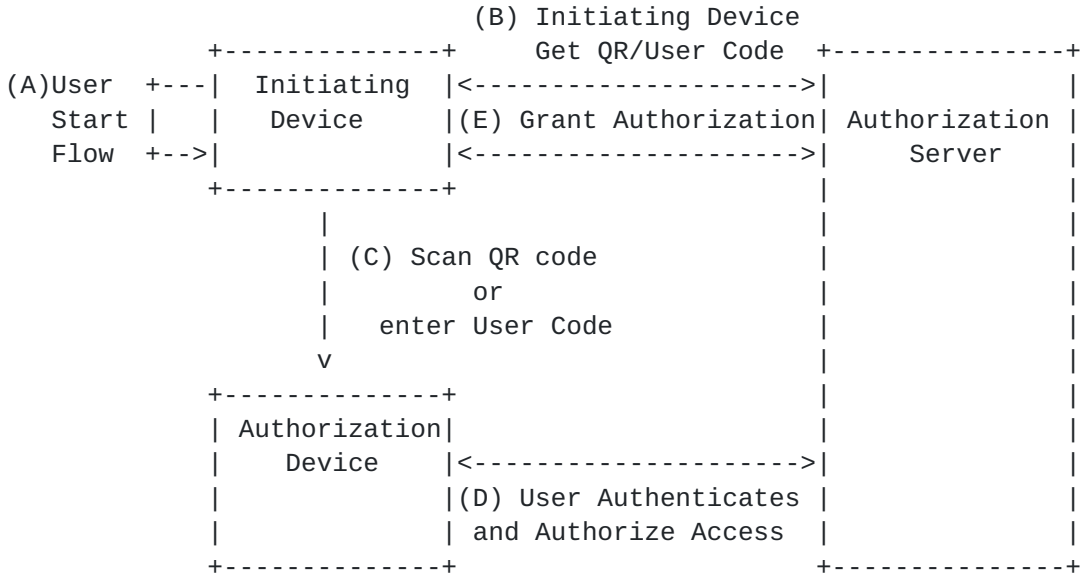

Figure 1: Cross Device Flows (User Transferred)

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

*(B) The initiating device retrieves a QR code or user code from an authorization server.

*(C) The QR code or user code is displayed on the initiating device where the user scans the QR code or enters the user code on the authorization device.

*(D) The user authenticates to the authorization server before granting authorization.

*(E) The Authorization Server issues tokens or grants authorization to the initiating device to access the user's resources.

The Device Authorization Grant ([RFC8628]) follows this pattern.

2.2. Client Transferred Pattern

The figure below shows an example of the client requesting the authorization server to initiate an authorization on the user's authorization device via the backchannel.

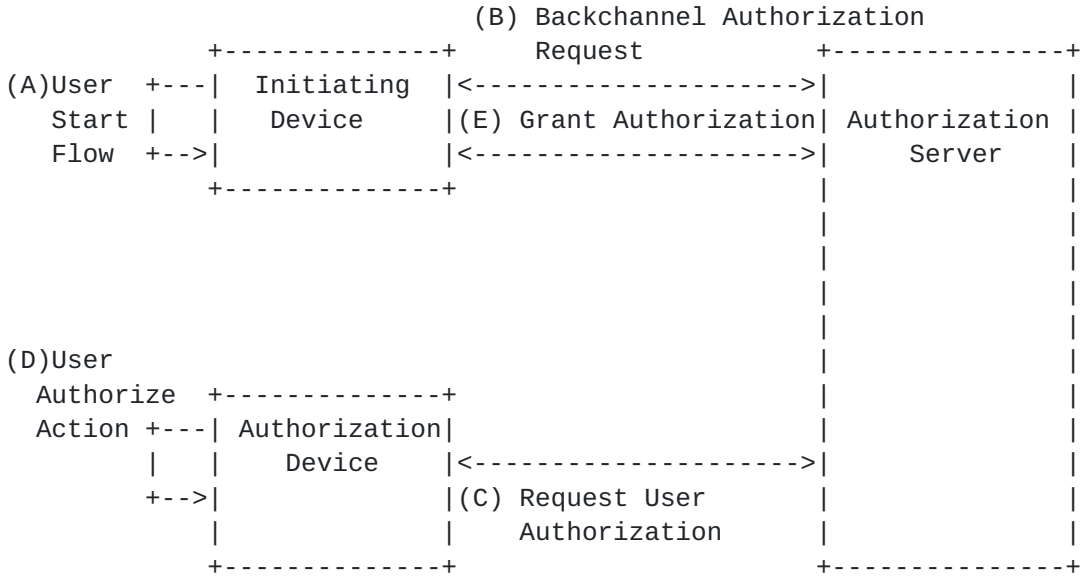

Figure 2: Cross Device Flows (Client Transferred)

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

*(B) The client on the initiating device requests user authorization on the backchannel from the authorization server.

*(C) The authorization server requests the authorization from the user on the user's device.

*(D) The user authenticates to the authorization server before granting authorization on their device.

*(E) The Authorization Server issues tokens or grants authorization to the initiating 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 and action.

The Client Initiated Backchannel Authentication [CIBA] follows this pattern.

2.3. Hybrid Pattern

The figure below shows an example of the client requesting the authorization server to initiate an authorization request via the backchannel.

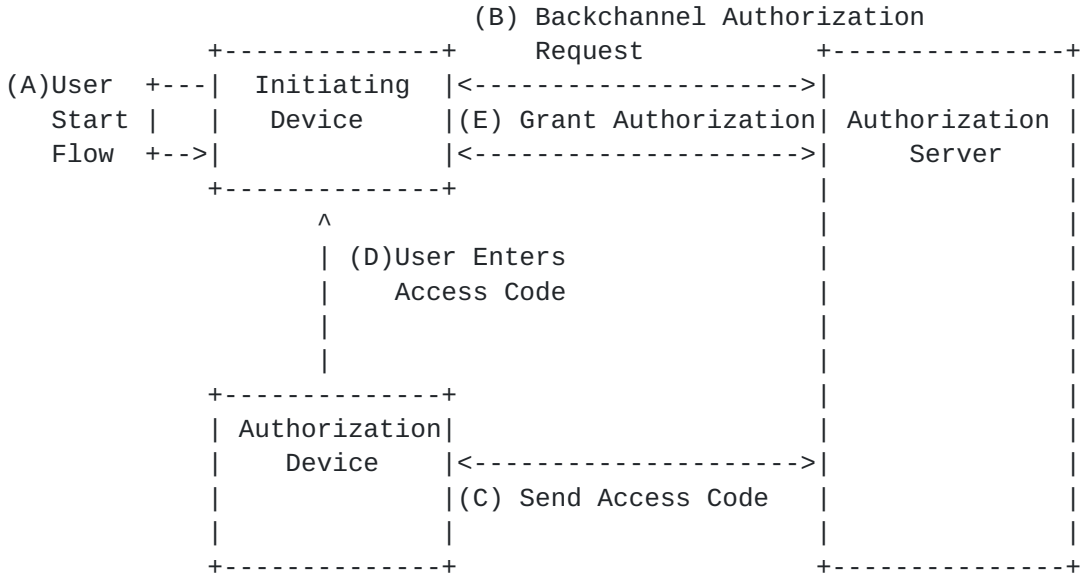

Figure 3: Cross Device Flows (Hybrid)

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

*(B) The client on the initiating device requests user authorization on the backchannel from the authorization server.

*(C) The authorization server sends an access code to the Authorization Device.

*(D) The user enters the access code on the Initiating Device.

*(E) The Authorization Server issues tokens or grants authorization to the initiating device to access the user's resources.

The Authorization Server may choose to authenticate the user before sending the access code. The access code may be delivered as a text message or through a mobile application.

2.4. Examples of cross-device flows

Examples of cross-device flow scenarios include:

2.4.1. Example A1: Authorize access to a video streaming service (User Transfer)

An end-user sets up a new smart TV and wants to connect it to their favorite streaming service. The TV displays a QR code 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.

2.4.2. Example A2: Authorize access to productivity services (User Transfer)

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.

2.4.3. Example A3: Authorize use of a bike sharing scheme (User Transfer)

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

2.4.4. Example A4: Authorize a financial transaction (Client Transfer)

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.

2.4.5. Example A5: Add a device to a network (Hybrid)

An employee is issued with 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 e-mail). The personal computer displays a QR code, which the employee scans with their mobile phone. The mobile phone is joined to the network and the employee can start accessing corporate data and services on their mobile device.

2.4.6. Example A6: Remote onboarding (User Transfer)

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 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.

2.4.7. Example A7: Transfer a session (Hybrid)

An employee is signed into 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 into the application on the mobile phone.

2.4.8. Example A8: Access a productivity application (Hybrid)

A user is accessing a Computer Aid Design (CAD) application. When accessing the application, an access code is sent to the user's mobile phone. The user views the access code on their phone and enters it in the CAD application, after which the CAD application displays the user's most recent designs.

3. Cross-Device Flow Exploits

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

3.1. User Transferred Pattern

A common action in cross-device flows is to present the user with a QR code or a user code on the initiating device (e.g., Smart TV) which is then scanned or entered on the authorization device (the 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's judgment to decide on whether they can trust the QR code or user code. In effect the user is asked to compensate for the absence of an authenticated channel between the initiating device (smart TV) and the device on which the authentication/ authorization will take place (the 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 trick end-users to scan the QR code or enter it on their mobile devices. Once the end-user performs the authorization on the mobile device, the attacker who initiated the authentication

or authorization request obtains access to the users resources. The figure below shows an example of such an attack.

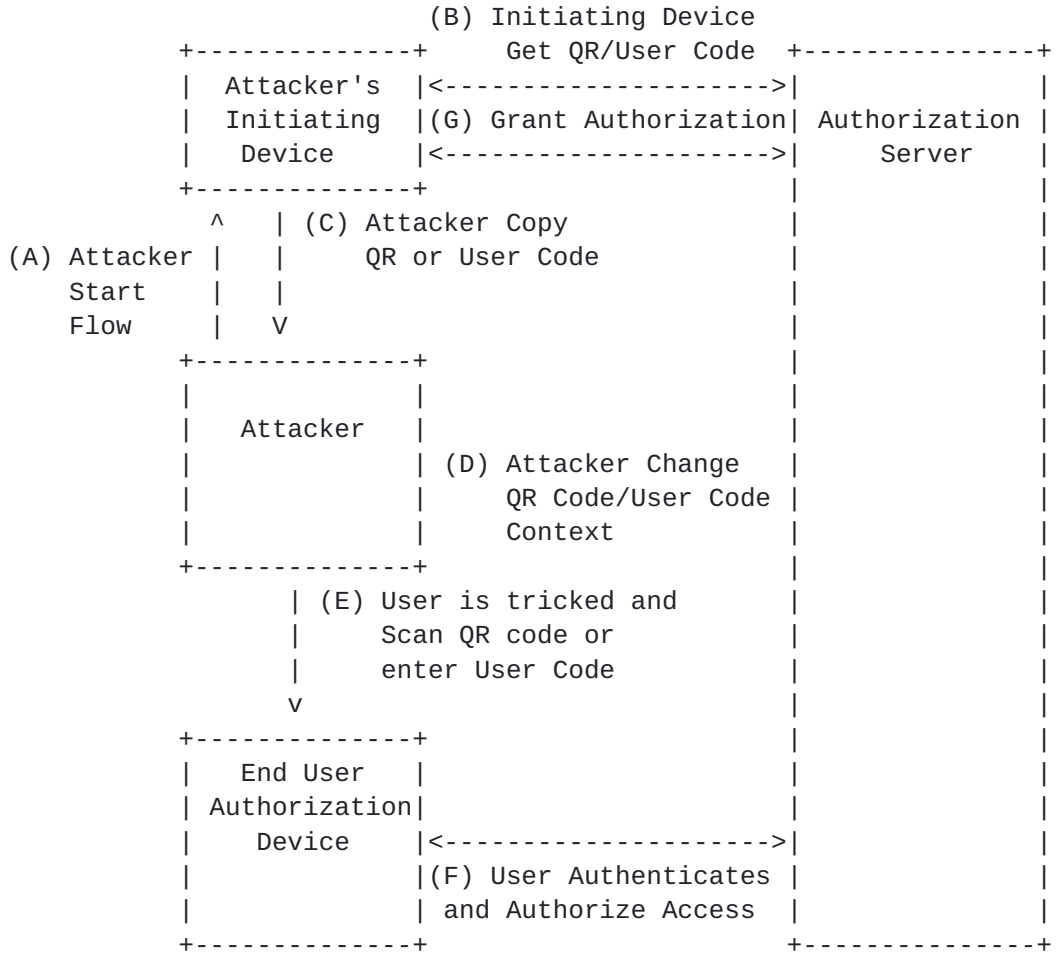

Figure 4: Attacker Initiated Cross Device Flow Exploit (User Transferred Pattern)

*(A) The attacker initiates the protocol on the initiating device (or by mimicking the initiating device) by starting a purchase, adding a device to a network or connecting a service to the initiating device.

*(B) The initiating device retrieves a QR code or user code from an authorization server

*(C) The attacker copies the QR code or user code

*(D) 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.

*(E) The QR code or user code is displayed in a context chosen by the attacker and the user is tricked into scanning the QR code or enter the user code on the authorization device.

*(F) The user authenticates to the Authorization Server before granting authorization.

*(G) The Authorization Server issues tokens or grants authorization to the initiating device, which is under the attackers control, to access the users resources and the attacker gains access to the resources and possibly any authorization artefacts like access and refresh tokens.

3.2. Client Transferred Pattern

In the client transferred pattern, the client instructs the authorization server to authetnicate the user and obtain authorization for an action. This may happen as a result of user interaction with the initiating device, but may also be triggered without the users direct interaction with the initiating 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 trick them into granting authroization. The social engineering techniques range in sophistication from messages misrepresenting the reason for receiving an authroizations requests through to triggering a large volume of requests at an inconvenient time for the user, in the hope that the user will grant authroization to make the requests stop. The figure below shows an example of such an attack.

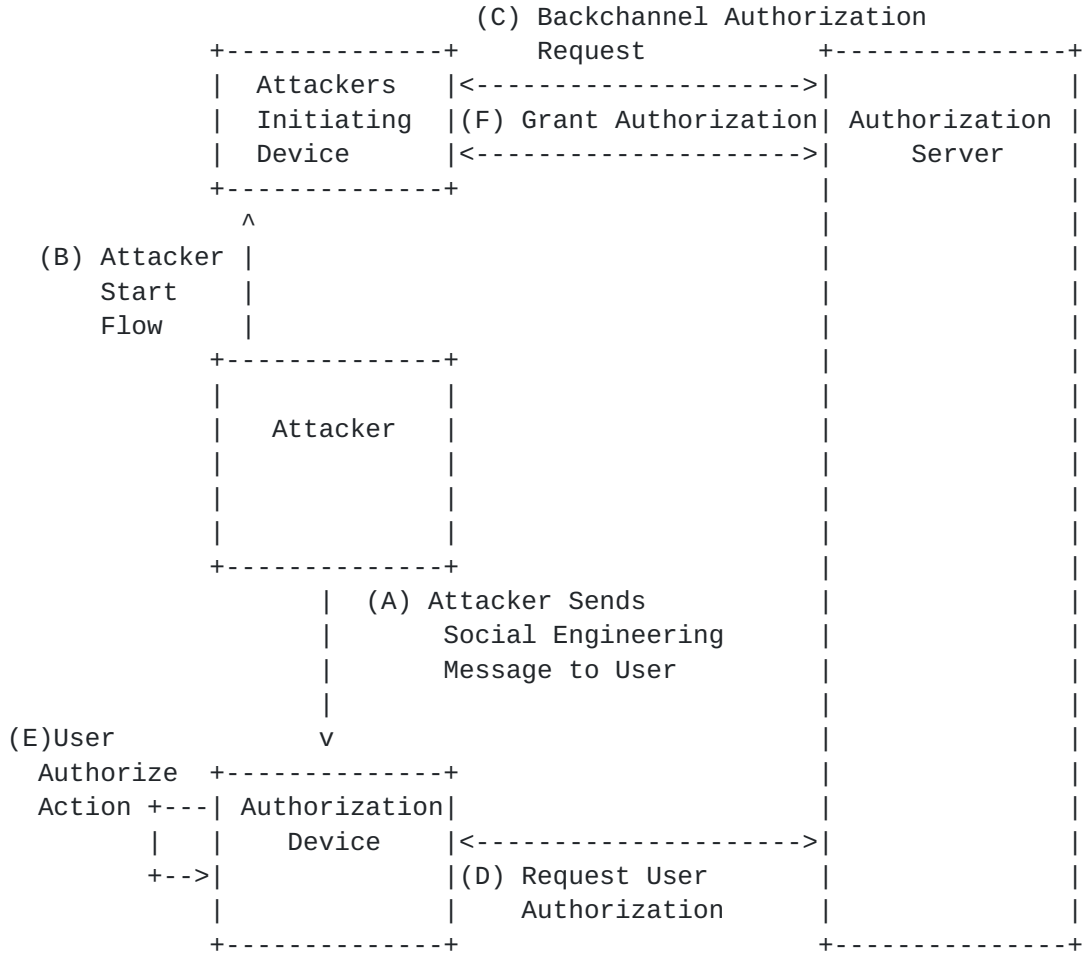

Figure 5: Attacker Initiated Cross Device Flow Exploit (Client Transferred Pattern)

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

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

*(C) The client on the initiating device requests user authorization on the backchannel from the authorization server.

*(D) The authorization server requests the authorization from the user on the user's device.

*(E) The user authenticates to the authorization server before granting authorization on their device.

*(G) The Authorization Server issues tokens or grants authorization to the initiating device, which is under the attackers control. The attacker gains access to the users

resources and possibly any authorization artefacts like access and refresh tokens.

3.3. Hybrid Pattern

In cross device flows that follow the Hybrid Pattern, the client initiates the authorizations request, but the user stil has to transfer the authorization code to the inititing device. The authorization request may happen as a result of user interaction with the initiating device, but may also be triggered without the users direct interaction with the initiating device.

Attackers exploit the hybrid pattern by combining the social engineering techniques used to set context for users and tricking users into providing them with access codes sent to their phones. These attacks are very similar to phishing attacks, except that the attacker also has the ability to trigger the authroization request to be sent to the user directly by the Authorization server.

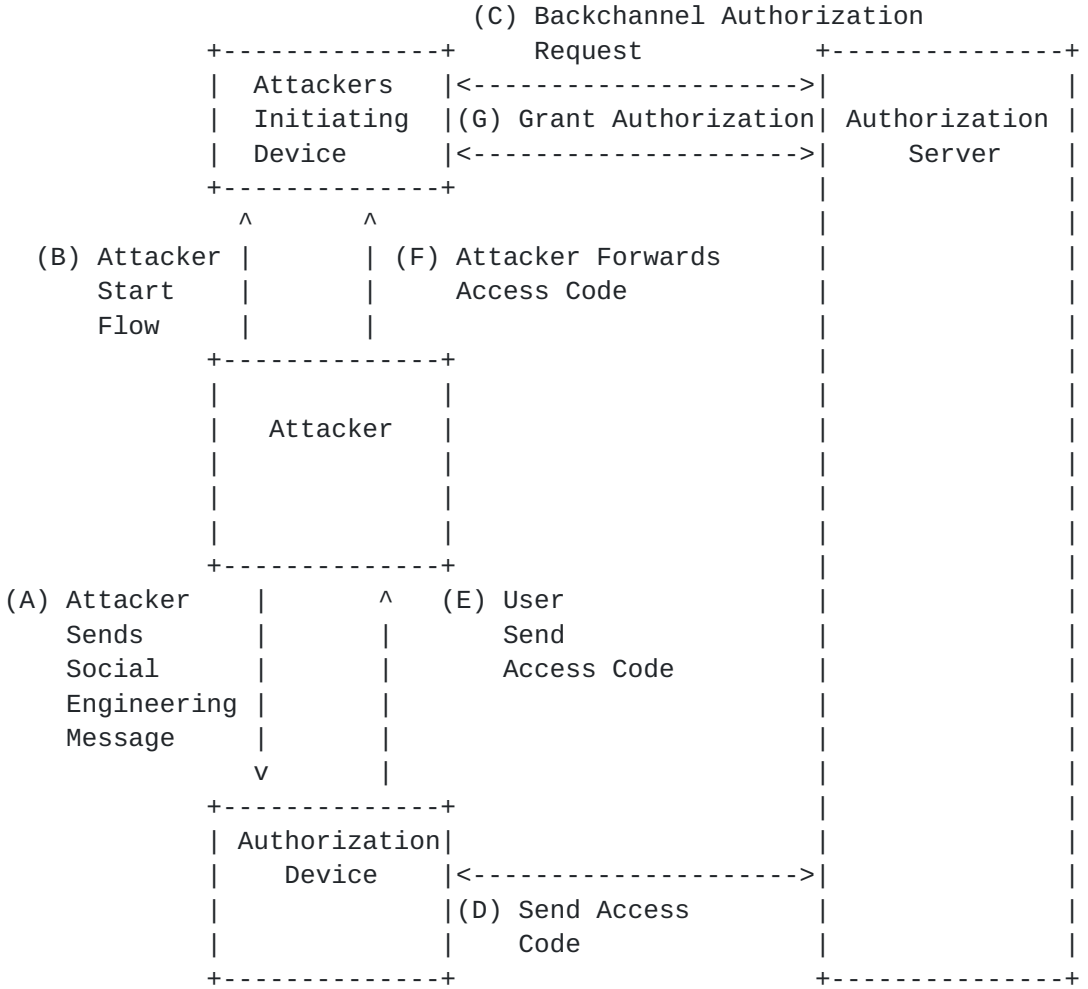

Figure 6: Attacker Initiated Cross Device Flow Exploit (Hybrid Pattern)

*(A) The attacker sends a social engineering message to prime the user for the authorization request they just received, along with instructions on what to do with the access code they received.

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

*(C) The client on the initiating device requests user authorization on the backchannel from the authorization server.

*(D) The authorization server sends an access code to the user's device (the access code may be presented as a QR code, or text message).

*(E) The user sends the access code to the attacker.

*(F) The attacker enters the access code on the Initiating Device.

*(G) The Authorization Server issues tokens or grants authorization to the initiating device, which is under the attackers control. The attacker gains access to the users resources and possibly any authorization artefacts like access and refresh tokens.

The unauthenticated channel may also be exploited in variations of the above scenario where the user initiates the flow and is then tricked into sending the QR code or user code to the attacker. In these flows, the user is already authenticated and they request a QR code or user code to transfer a session or obtain some other privilege such as joining a device to a network. The attacker then proceeds to exploit the unauthenticated channel by using social engineering techniques to trick the user into initiating a flow and send the QR code or user code to the attacker, which they can then use to obtain the privileges that would have been assigned to the user.

3.4. Examples of cross-device flow exploits

The following examples illustrate these attacks in practical settings and 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 into granting consent to avail of services, access data and make payments.

3.5. Example B1: Illicit access to a video streaming service (User Transferred Pattern)

An attacker obtains a smart TV and attempts to access an online streaming service. The smart TV obtains a QR code from the

authorization server and displays it on screen. The attacker copies the QR code and embeds it in an e-mail that is sent to a large number of recipients. The e-mail 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 open the e-mail and scan the QR code to register for early access to premium content. The users perform 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.

3.6. Example B2: Illicit access to productivity services (User Transferred Pattern)

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 e-mail 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 e-mail 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. Once these employees authorized access, the attacker obtains access and refresh tokens from the authorization server and uses it 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.

3.7. Example B3: Illicit access to physical assets (User Transferred Pattern)

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

another bicycle and lodges a complaint with the bike renting company.

3.8. Example B4: Illicit Transaction Authorization (Client Transferred 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.

3.9. Example B5: Illicit Network Join (Hybrid Pattern)

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.

3.10. Example B6: Illicit Onboarding (User Transferred Pattern)

An attacker initiates an employee onboarding flow and obtains a QR code from the onboarding portal to invoke a 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 e-mail 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 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.

3.11. Example B7: Illicit session transfer (Hybrid Pattern)

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 and obtain a QR code and then send the QR code to the attacker. The attacker scans the QR code with their mobile phone and access the users data and resources.

3.12. Example B8: Account takeover (User Transferred Pattern)

An attacker wants to use some website which requires presentation of a verifiable credential for authentication. The attacker creates a phishing website which will in real time capture log-in QR Codes from the original website and present these to the victim. The attacker tries to get the victim to use the phishing website using an e-mail campaign etc. The victim scans the QR code on the phishing website, invokes their wallet and presents their credentials. Once the credentials are presented, the original session from the attackers device is authenticated with the victim's credentials.

3.13. Out of Scope

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

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

4. 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. a. E.g. the attacker can obtain a QR code or user code, or can request an authentication/ authorization decision from the user.

2. The interaction between the initiating device and authentication device is unauthenticated. a. E.g. it is left ot the user to decide if the QR code, user code or authentication request is being presented in a legitimate context

A number of 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 (smart TVs, printers, kiosks). In this protocol, the user code or QR code is displayed on the initiating device and entered on a second device (e.g., a mobile phone).

*Open ID Foundation Client Initiated Back-Channel Authentication (CIBA) [CIBA]: A standard developed in the OpenID Foundation that allows a device or service (e.g., a personal computer, Smart TV, 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 initiating device, but is instead contacted by the OpenID Provider to complete the authentication using a push notification, e-mail, 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) or present credentials (Presentation). The QR code is presented on a device that initiates the flow.

* Self-Issued OpenID Provider v2 (SIOP V2): A standard that allows end-user to present self-attested or third party attested attributes when used with Opend ID 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 initiating device and authorization device is 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]), the cross-device mode should not be used for same-device scenarios. If an implementor 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.

5. Mitigating Against Cross-Device Flow Attacks

The unauthenticated channel between the initiating device and the authenticating 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" and 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 the exploits of cross-device flows require a three-pronged approach:

1. Secure deployed protocols with practical mitigations.

2. Adopt or develop more secure protocols where possible.

3. Provide analytical tools to assess vulnerabilities and effectiveness of mitigations.

5.1. Practical Mitigations

A number of protocols that enable cross-device flows that are susceptible to illicit consent grant 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 are applied whenever implementing a cross-device flow. Every mitigation provides an additional layer of security that makes it harder to initiate the attack, disrupts attacks when in process or reduces the impact of a successful attack.

5.1.1. Establish Proximity

The unauthenticated channel between the initiating and authenticating device allows attackers to obtain a QR code or user code in one location and display in another location. Establishing proximity between the location of the initiating device and the authentication device limits an attacker's ability to launch attacks by sending the user or QR codes to large numbers of users across the globe. There are a couple of ways to establish proximity:

*Physical connectivity: This is a good indicator of proximity, but requires specific ports, cables and hardware and 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.

*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.

* 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 (initiating device) with that of the authentication/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 initiating 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.

* Geo-location: Proximity can be established by comparing geo-location information derived from global navigation satellite-system (GNSS) co-ordinates or geolocation lookup of IP addresses and comparing proximity. Due to inaccuracies, this 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 users device, provided that the information encoded in a QR code is integrity protected using a digital signature.

Dependig on the risk profile and the threat model in which as system is operating, it mey be neccesary to user mor than one mechanism to establish proximity to raise the bar for any potential attackers.

Note: There are scenarios that require that an authorization takes place in a different location than the one in which the transaction is authorized. 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 may be an indicator of risk and the system may deploy additional controls (e.g., transaction value limits, transaction velocity limits) or use the proximity information as input to a risk management system.

Limitations: Proximity mechanisms raises the bar for an attack. 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 his initiating device using standard debugging features available on many platforms.

5.1.2. Short Lived/Timebound User Codes

The impact of an attack can be reduced by making 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.

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, or incorrectly entering a code).

5.1.3. 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.

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. As a result, it may be practical to allow the same code to be presented a small number of times.

5.1.4. 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 authorizations 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 eqaul to lifetime of a token if short lived/timebound tokens are used (see Short Lived/Timebound Codes). This will limit the size of the deny list.

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.

5.1.5. 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 (e-mail, text, instant messaging etc). 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 being reused in multiple messages to disrupt spray attacks.

Limitations: Content filtering may be better suited to interrupt large scale spray attacks since some scenarios may require re-transmission of user, QR and access codes. Content filtering may also be fragmented across multiple communications systems and channels (e-mail, messaging, text etc), making it harder to detect or interrupt attacks that are executed over multiple channels, unless here is a high degree of integration between content filtering systems.

5.1.6. Detect and remediate

The authorization server may be able to detect misuse of the codes due to repeated use as described in Unique Codes, as an input from a content filtering engine as described in Content Filtering, 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 Continious Access Evaluation Protocol (CAEP) messages [CAEP] using the Shared Signals and Events (SSE) [SSE] framework or an equivalent notification system.

Limitations: Detection and remediation requires 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.

5.1.7. 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 by mimicking devices to initiate the protocol. Trusted devices include devices that are pre-registered with the authorization server or are subject to device management policies. Device management policies may enforce 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 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.

Limitations: An attacker may still be able to obain access to a trusted device and use it to initiate authorization requests, making it necesary to apply additional controls and integrating with other threat detection and management systems that can detect suspicious behaviour such as repeated requests to initiate authorization or high volume of service activation on the same device.

5.1.8. 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. By limiting protocol initiation to a specific network, the attacker needs to have access to a device on the network.

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, it should be used in concert with other controls outlined in this document to achieve defence in-depth.

5.1.9. 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 are 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 initiating device.

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.

5.1.10. 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.

Limitations: Short lived tokens reduces the time window during which an attacker can benefit from a succesfull 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.

5.1.11. Rate Limits

An attacker that engages in a scaled spray attack needs to request a large number of user codes (see exploit Example B1) or initiate a large number of authorization requests (see exploit Example B4) in a short period of time. An authorization server can apply rate limits to minimize the number of requests it would accept from a client in a limited time period.

Limitations: Rate limits are effective at slowing an attacker down and helps to degrade spray attacks, but does not prevent more targeted attacks that are executed with lower volumes and velocity. Therefor it should be used along with other techniques to provide a defence-in-depth against cross-device attacks.

5.1.12. 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 an 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 for sender constraining 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-posession, raises the bar for executing the attack and and profiting from exfiltrating tokens. The quality of key protection has an impact on the effectiveness of the attack, and although a software proof-of-posession key is better than no proof-of-posession key, an attacker may still exfiltrate the software key. Hardware keys will be harder to exfiltrate, but comes with aditional implementation complexity. An attacker that controls the initiating device may still be able to excercise they key, even if it is in hardware, thus the main protection derived from sender constrianed tokens is preventing them from being moved from the iniating device to another device that can be used to profit from the attack.

5.1.13. 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. The user experience should reinforce the message that unless the user initiated the authorization request, or was expecting it, they should decline the request.

It should be clear to the user how to decline the request. To avoid accidental authorization grants, the "decline" option may be the default option or given similar prominence in the user experience as the "grant" option.

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 service may provide out-of-band reinforcement to the user on the context and conditions under which an authorization grant may be requested. For example if the service provider does not send e-mails with QR codes requesting users to grant authorization, this may be reinforced in marketing messages, in-app experiences and through anti-fraud awareness campaigns.

Limitations: Improvements to user experience on their own is unlikely to be sufficient and should be used in conjuntion with other controls described in this document.

5.1.14. Authenticated flow

By requiring a user to authenticate on the initiating 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. This prevents the attacker from obtaining a QR code or user code that they can use to mislead an unsuspecting user. This requires that the initiating 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: Starting with and authenticated does not prevent the attacks described in Example B5: Illicit Network Join and Example B7: Illicit Session Transfer and it is recommended that additional mitigations described in this document is used if the cross-device flows are used in scenarios such as Example A5: Add a device to a network and Example A7: Transfer a session.

5.1.15. Practical Mitigation Summary

The practical mitigations described in this section can prevent the attacks from being initiated, disrupt attacks once they start or reduce the impact or remediate an attack if it succeeds. When combining one or more of these mitigations the overall security profile of a cross-device flow improves significantly. The following table provides a summary view of these mitigations:

Mitigation	Prevent	Disrupt	Recover
Establish Proximity	X	X	
Short Lived/Timebound 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 Experience	X		
Authenticated flow	X		

Table 1: Practical Mitigation Summary

5.2. 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.

5.2.1. IETF OAuth 2.0 Device Authorization Grant *[RFC8628]*:

5.2.1.1. Description

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

5.2.1.2. Susceptibility

There are several reports in the public domain outlining how the unauthenticated channel may be exploited to execute an illicit consent grant attack.

5.2.1.3. 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.

5.2.1.4. Mitigations

In addition to the security considerations section in the standard, 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.

5.2.1.5. 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 initiating device and authorization device is the same device).

5.2.2. OpenID Foundation Client Initiated Back-Channel Authentication (CIBA):

5.2.2.1. Description

Client Initiated Back-Channel Authentication (CIBA) [CIBA]: A standard developed in the OpenID Foundation that allows a device or service (e.g., a personal computer, Smart TV, 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 initiating device, but is instead contacted by the OpenID Provider to complete the authentication using a push notification, e-mail, text message or any other suitable mechanism.

5.2.2.2. Susceptibility

Less susceptible to unauthenticated channel attacks, but still vulnerable to attackers who know or can guess the user identifier and initiate a spray attack as described in Example 4.

5.2.2.3. Device capabilities

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

5.2.2.4. Mitigations

In addition to the security considerations section in the standard, 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.

5.2.2.5. When to use

Use CIBA instead of Device Authorization Grant if it is possible for the initiating device to obtain a user identifier on the initiating 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 initiating device and authorization device is the same device).

5.2.3. FIDO2/WebAuthn

5.2.3.1. 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", which enables an external device, such as a phone or tablet, to be used as a roaming authenticator for signing into the primary device, such as a personal computer. This is commonly called FIDO Cross-Device Authentication (CDA).

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 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.

5.2.3.2. Susceptibility

The Cross-Device Authentication flow proves proximity by leveraging BLE advertisements for service establishment, significantly reducing the susceptibility to any of the exploits described in Examples 1-6.

5.2.3.3. Device capabilities

Both the initiating device and the authenticator require BLE support. The initiating device must support both FIDO2/WebAuthn, specifically CTAP 2.2 with hybrid transport. The mobile phone must support CTAP 2.2+ to be used as a cross-device authenticator.

5.2.3.4. Mitigations

FIDO Cross-Device Authentication (CDA) establishes proximity through the use of BLE, reducing the need for additional mitigations. An implementer may still choose to implement additional mitigation as described in this document.

5.2.3.5. When to use

FIDO2/WebAuthn should be used for cross-device authentication scenarios whenever the devices are capable of doing so. It may be used as an authentication method with the Authorization Code Grant [RFC6749] and PKCE [RFC7663], to grant authorization to an initiating device (e.g., Smart TV or interactive whiteboard) using a mobile phone as the authenticating device. 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].

5.2.4. Protocol Selection Summary

The FIDO Cross-Device Authentication (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, Client Initiated Backchannel Authentication (CIBA) provides an alternative, provided that there is a channel through which the authorizations server can contact the end user. Examples of such a channel include device push notifications, e-mail or text messages which 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 is only used when additional mitigations are deployed to prevent attacks that exploit the unauthenticated channel between devices.

5.3. 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, expressiveness), in the kind of statements they can produce, and whether the proofs can be assisted by tools or have to be performed manually. One of the most successfully used approaches is the so-called Web Infrastructure Model (WIM), a model 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 in unprecedented detail. Using the WIM, previously unknown flaws in OAuth, OpenID Connect, and FAPI were discovered.

To ensure secure cross-device interactions, a formal analysis using the WIM therefore seems to be in order. Such an analysis should comprise a generic model for cross-device flows, potentially including different kinds of interactions. The aim of the analysis would be to evaluate the effectiveness of selected mitigation strategies. To the best of our knowledge, this would be the first study of this kind.

6. 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 towards 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 initiating and authentication/authorizing device using techniques commonly used in phishing attacks. These 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 the deployment of practical mitigations to safeguard protocols that are already deployed, provide guidance on when to use different protocols, including protocols that are not susceptible to these attacks, and the introduction of formal methods to evaluate the impact of mitigations and find additional issues.

7. Contributors

We 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 and others (please let us know, if you've been mistakenly omitted) for their valuable input, feedback and general support of this work.

8. Informative References

[CAEP] Tulshibagwale, A. and T. Cappalli, "OpenID Continuous Access Evaluation Profile 1.0 - draft 01", June 2021, <https://openid.net/specs/openid-caep-specification-1_0-01.html>.

[CIBA] Fernandez, G., 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>.

[Exploit1] Cooke, B., "The Art of the Device Code Phish", July 2021, <https://0xboku.com/2021/07/12/ArtOfDeviceCodePhish.html>.

[Exploit2] "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>.

[Exploit3] Syynimaa, N., "Introducing a new phishing technique for compromising Office 365 accounts", October 2020,

<https://o365blog.com/post/phishing/#new-phishing-technique-device-code-authentication>.

[Exploit4] Hwong, J., "New Phishing Attacks Exploiting OAuth Authentication Flows (DEFCON 29)", August 2021, <https://www.youtube.com/watch?v=9slRYvpKHp4>.

[Exploit5] "OAuth's Device Code Flow Abused in Phishing Attacks", August 2021, <https://www.secureworks.com/blog/oauths-device-code-flow-abused-in-phishing-attacks>.

[Exploit6] "SquarePhish: Advanced phishing tool combines QR codes and OAuth 2.0 device code flow", August 2022, <https://www.helpnetsecurity.com/2022/08/11/squarephish-video/>.

[NYC.Bike] Byrne, K.J., "Citi Bikes being swiped by joyriding scammers who have cracked the QR code", 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., Medeiros, B.d., and C. Mortimore, "OpenID Connect Core 1.0", November 2014, <http://openid.net/specs/openid-connect-core-1_0.html>.

[OpenID.SIOPV2] Yasuda, K., Jones, M., and T. Lodderstedt, "Self-Issued OpenID Provider v2", November 2022, <https://bitbucket.org/openid/connect/src/master/openid-connect-self-issued-v2/openid-connect-self-issued-v2-1_0.md>.

[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>.

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

[RFC7663] Trammell, B., Ed. and M. Kuehlewind, Ed., "Report from the IAB Workshop on Stack Evolution in a Middlebox Internet (SEMI)", RFC 7663, DOI 10.17487/RFC7663, October 2015, <https://www.rfc-editor.org/info/rfc7663>.

[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>.

[SSE] Tulshibagwale, A., Cappalli, T., Scurtescu, M., Backman, A., and J. Bradley, "OpenID Shared Signals and Events Framework Specification 1.0", June 2021, <https://openid.net/specs/openid-sse-framework-1_0-01.html>.

Appendix A. Document History

[[To be removed from the final specification]]

-01

* Added additional diagrams and descriptions to distinguish between different cross-device flow patterns.

* Added short description on limitations of each mitiagtion.

* Added acknowledgement of additional contributors.

* Fixed document history format.

-00 (Working Group Draft)

* Initial WG revision (content unchanged from draft-kasselman-cross-device-security-03)

-03 draft-kasselman-cross-device-security

* Minor edits and typos

-02 draft-kasselman-cross-device-security

* Minor edits and typos

* Upload as draft-ietf-oauth-cross-device-security-best-practice-02

-01 draft-kasselman-cross-device-security

* Updated draft based on feedback from version circulated to OAuth working group

* Upload as draft-ietf-oauth-cross-device-security-best-practice-01

-00 draft-kasselman-cross-device-security

* Initial draft adopted from document circulated to the OAuth Security Workshop Slack Channel

* Upload as draft-ietf-oauth-cross-device-security-best-practice-00

Authors' Addresses

Pieter Kasselman
Microsoft

Email: pieter.kasselman@microsoft.com

Daniel Fett
yes.com

Email: mail@danielfett.de

Filip Skokan
Okta

Email: panva.ip@gmail.com