Cross-Device Flows (Draft)

Threats and Mitigations

Purpose

This document describes the 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 guide to system designers, architects, product managers, security specialists, fraud analysts and engineers implementing cross-device flows.

Overview

Cross-device flows enable a user to start a scenario on one device (e.g. a PC) and then use a second, personally trusted device (e.g. a smartphone), to authenticate and authorize access to a resource (e.g. access to a service). These flows are increasingly popular and are often initiated by using a mobile phone to scan a QR code or enter a user code displayed on an initiating device (e.g. Smart TV, Kiosk, PC etc). In other variants of these flows, the user takes an action on the initiating device (e.g. initiates a purchase or adds a device to a network) which triggers an event on their mobile phone, requesting them to authenticate or authorize a transaction.

The channel between the initiating device and the authentication/authorization device is unauthenticated and relies on the user's judgment to decide whether to trust a QR code, user code or authorization request. Several publications have emerged in the public domain, 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. They skip the step of collecting credentials by persuading users into granting authorization to access resources and then collecting the access and refresh tokens. These attacks are effective, even when multi-factor authentication is deployed.

In response to these attacks, we propose a three pronged approach that includes:

1. Deploying practical mitigations with existing protocols.
2. Selecting protocols that are not susceptible to unauthenticated channel exploits.
3. Formal analysis of cross-device flows that can be used to assess the effectiveness of the proposed mitigations.

What is a Cross-Device Flow?

In a cross-device flow, a user starts a scenario on one device (e.g. a PC) and then uses a second device (e.g. a smartphone) to authenticate and authorize access to a resource (e.g. access to a service). This has 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 PC).
- **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.

Examples of cross-device flow scenarios include:

Example 1: Authorize access to a video streaming service

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 grant authorization, after which the streaming service is available on the smart TV.

Example 2: Authorize access to productivity services

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 PC 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.

Example 3: Authorize use of a bike sharing scheme

An end-user wants to rent a bicycle from a bike sharing scheme. The bicycles are locked into 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.

Example 4: Authorize a financial transaction

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.

Example 5: Add a device to a network.

An employee is issued with a laptop computer that is already joined to a network. The employee wants to add their mobile phone to the network to allow them to access corporate data and services (e.g. files and e-mail). The PC 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.

Example 6: Remote onboarding

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 mobile 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.

Cross-Device Flow Exploits

The benefits of cross-device flows is compelling and is seeing adoption for a range of consumer and enterprise scenarios (see Scenarios). To ensure the user and service provider enjoy the benefits of using their mobile phones as authentication and authorization devices, the interaction between the two devices needs to be secure.

A common action in these cross-device flows is to present the user with a QR code or a user code on the initiating device (e.g. Smart TV) and scanned or entered on the second 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 harvesting QR codes or user codes and then using social engineering techniques to trick end-users to scan the QR code or enter it on their mobile devices. The end-user performs the authentication/authorization on the mobile device, and as a result the attacker who initiated the authentication or authorization request obtains access to the users resources. These attacks are also known as illicit consent grant attacks.

The following examples illustrate these attacks in practical settings and shows how the unauthenticated channel is exploited by attackers who can copy the QR codes and user codes, change the context in which it is presented using social engineering techniques and misleads end-users into granting consent to avail of services, access data and make payments.

Example 1: Illicit access to a video streaming service

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. They 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.

Example 2: Illicit access to productivity services

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.

Example 3: Illicit access to physical assets

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. The customer proceeds to unlock another bicycle and lodges a complaint with the bike renting company.

Example 4: Illicit Transaction Authorization

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.

Example 5: Illicit Network Join

An employee that is colluding with an attacker logs into their PC and initiates the process to add a new device to the network. They authenticate to the network and obtain a QR code. They send the QR code to the attacker they are colluding with. 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.

Example 6: Illicit Onboarding

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 obtained the QR code displays a message to the attacker with instructions on how to access their account.

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 (RFC 8682): 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): A standard under development as part of the Financial-grade API (FAPI) family of standards that allows a device or service (e.g. a PC, 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.

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.

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” (i.e. experts in). They are 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, the exploits of cross-device flows require a three-pronged approach that:

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.

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. Prevent the attack from being initiated.
2. Disrupts the attack once it is initiated.
3. Remediate or reduce 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.

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 GPS 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.

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.

Short Lived/Timebound Codes

The impact of an attack can be reduced by making codes short lived. If an attacker obtains a short-lived token, it limits the duration during which the unauthenticated channel can be exploited, which increases the cost of a successful attack.

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 QR code which is validated by the authorization server when the user scans the QR code.

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 or deploy other risk mitigations if it detects the same codes being presented repeatedly.

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.

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.

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. By including limited scopes, it 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.

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 does not prevent or disrupt the attack, but serves as a remedial mechanism in case the attack succeeded.

Rate Limits

An attacker that engages in a scaled spray attack needs to request a large number of user codes (see exploit example 1) or initiate a large number of authorization requests (see exploit example 2) 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.

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.

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 should also be offered an option to decline the request, or be made aware that unless they initiated the request, they should decline it.
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 our files, you should decline this request and notify the security department").

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:

	Prevent	Disrupt	Recover
Establish Proximity	X	X	
Short Lived/Timebound Codes		X	
Content Filtering		X	

Trusted Devices	X		
Limited Scopes			X
Short Lived Tokens			X
Rate Limits	X	X	
Sender Constrained Tokens			X
User Experience	X		

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 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.

IETF OAuth 2.0 Device Authorization Grant (RFC 8682):

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

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

Description: A standard under development as part of the Financial-grade API (FAPI) family of standards that allows a device or service (e.g. a PC, 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.
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.
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.
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.
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.

FIDO2/WebAuthn

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 desktop or laptop. 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.

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.

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.

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.

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 (RFC 6749) and PKCE (RFC 7663), 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 (RFC 8628).

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, Channel Initiated Backchannel Authentication (CIBA) provides an alternative, provided that the underlying devices can receive push notifications. If CIBA is used, additional mitigations to enforce proximity and initiate transactions from trusted devices 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.

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.

Conclusion

Cross-device flows enable authorization on devices with limited input capabilities, allows for secure authentication when using public or shared devices, provides a path towards multi-factor authentication and provides 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 harvest 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.

Contributors

Pieter Kasselman (Microsoft), Daniel Fett (Yes.com), Filip Skokan (Okta), Tim Cappalli (Microsoft)