Coalition for Content Provenance and Authenticity

C2PA Security Considerations

1.2, 2024-07-02: Release

Table of Contents

Chapter 1. Introduction

This document provides information security considerations for technology described in the C2PA core specification. It complements security-related content in other C2PA documents. This document includes:

- A summary of relevant security features of C2PA technology
- Security considerations for practical use of C2PA technology
- Threats to C2PA technology and respective treatment of those threats, including countermeasures

Analysis in this document does not assess security risk or likelihoods associated with threats or outcomes. These may vary based on a number of factors, such as the context in which C2PA technology is deployed and changes in the threat landscape. However, content in this document may be helpful in developing security risk and likelihood assessments related to use of C2PA technology.

IMPORTANT Content in this document is non-normative and does not specify requirements for complying with the C2PA specification.

While the C2PA endeavours to develop and maintain a comprehensive threat model and security considerations for C2PA technology, we expect this work to be ongoing as technology develops and the threat landscape evolves.

IMPORTANT This document is a work in progress.

1.1. Threat modelling process overview

The C2PA builds security into our designs as they are being developed, but also expects that security design and threat modelling will continue as the system, ecosystem, and threat landscape evolve.

To this end, the C2PA uses a focused threat modelling process to support development of a strong security and privacy design. Outcomes of the effort directly support development of documentation about explicit threats and security considerations while also facilitating security-oriented thinking throughout the design process.

The threat modelling process combines synchronous (i.e., live) threat modelling sessions with focused groups of Subject Matter Experts (SMEs) in the asynchronous development of content. The number of attendees in each synchronous session is kept small to promote efficient discussions, but all members of the C2PA have the opportunity to participate via either modality.

Like other security activities, we expect our threat modelling process to evolve with the C2PA ecosystem. Process documentation is considered a guide rather than a strict directive on how threat modelling works within the C2PA.

NOTE In this document *asset* refers to C2PA assets and not conventional threat model assets.

1.1.1. References

A variety of references and experiences are used to inform threat modelling and related security activities for the

C2PA. This section provides a subset of public documents for reference.

- IETF on security considerations
- IETF on privacy considerations (guidelines)
- W3C security and privacy self-review questionnaire
- OAuth2 threat model (example)
- Threat Modelling: Designing for Security
- OWASP Threat Modelling
- Microsoft Threat Modelling

Chapter 2. Security features

This section summarizes C2PA features that support the security model specified later in this document. The C2PA core specification should be considered the canonical reference for C2PA features. It supersedes summaries provided here.

2.1. Provenance model

C2PA enables consumers to reason about asset provenance. C2PA specifies a model for asset provenance based on the following definitions.

Provenance

The logical concept of understanding the history of an *asset* and its interaction with *actors* and other *assets*, as represented by the *provenance data*.

Provenance data

The set of *C2PA Manifest_s for an _asset* and, in the case of a *composed asset*, its *ingredients*.

NOTE A *C2PA Manifest* can reference other _C2PA Manifest_s.

Assertion

A data structure which represents a statement asserted by an *actor* concerning the *asset*. This data is a part of the *C2PA Manifest*.

Claim

A digitally signed and tamper-evident data structure that references a set of *assertions* by one or more *actors*, concerning an *asset*, and the information necessary to represent the *content binding*. If any *assertions* were redacted, then a declaration to that effect is included. This data is a part of the *C2PA Manifest*.

Claim signature

The digital signature on the *claim* using the private key of an *actor*. The *claim signature* is a part of the *C2PA Manifest*.

C2PA Manifest

The set of information about the *provenance* of an *asset* based on the combination of one or more *assertions* (including *content bindings*), a single *claim*, and a *claim signature*. A *C2PA Manifest* is part of a *C2PA Manifest Store*.

NOTE A *C2PA Manifest* can reference other _C2PA Manifest_s.

Origin

The *C2PA Manifest* in the *provenance data* which represents the software or device that initially created the *asset*.

NOTE Details on how one determines which *C2PA Manifest* is the *origin* are left for specification.

Active Manifest

The last manifest in the list of *C2PA Manifests* inside of a *C2PA Manifest Store* which is the one with the set of *content bindings* that are able to be validated.

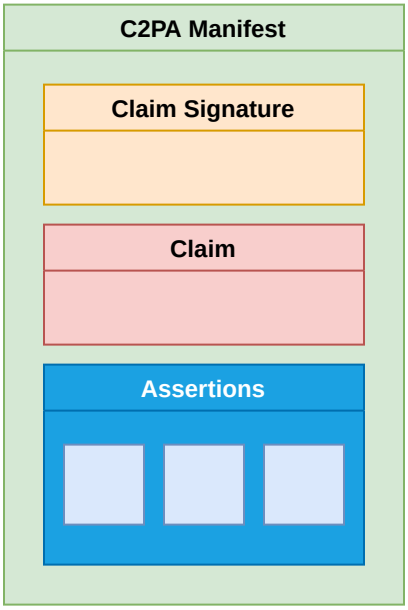

Figure 1. A C2PA Manifest and its constituent parts

2.2. Trust model

The basis of making trust decisions in C2PA, its Trust Model, is the identity of the actor associated with the cryptographic signing key used to sign the claim in the active manifest. The identity of a signatory is not necessarily a human actor, and the identity presented may be a pseudonym, completely anonymous, or pertain to a service or trusted hardware device with its own identity, including an application running inside such a service or trusted hardware. C2PA Manifests can be validated indefinitely regardless of whether the cryptographic credentials used to sign its contents are later expired or revoked.

In addition to identity, C2PA leverages trusted time-stamps to ensure the correctness of signing and revocation processes. A benefit of this approach is that revocation status can be optionally obtained when claims are generated (rather than when consumed), and revocation can be applied to subsets of signatures by time-stamp. See the Trust Model section of the specification for more details.

2.3. Claim signatures

C2PA uses a combination of cryptographic hashes and signatures to ensure that any tampering of the asset provenance can be detected. C2PA does not offer any protection against the complete removal of C2PA manifests from assets, but the design does aim to prevent any unauthorized tampering of either the digital asset content or the C2PA manifests themselves. The C2PA Guidance document for implementers offers direction about how soft bindings and manifest repositories could be used to address the removal situation.

Details on supported cryptographic hashing and signature schemes, as well as format details, are covered in the

specification's Cryptography section. As noted there, the C2PA maintains lists of supported algorithms and certificate types that aim to adhere to cryptographic and compatibility best practices.

Detail on how signing credentials relate to identity are included in the Trust Model section of the specification. The Validation section provides details on validating claims, which includes validating signatures.

2.4. Content bindings

Content bindings associate provenance data in C2PA Manifests with assets.

In the context of a C2PA Manifest, hard bindings provide a strict cryptographic binding between provenance and an asset; they aim to provide some useful guarantees against tampering. Soft bindings provide less strict binding between the provenance and an asset. They may associate C2PA provenance with multiple assets, and vice versa. The soft binding section goes into details about how soft bindings can be used with C2PA Manifests.

C2PA Manifest Stores are usually embedded into assets but they may exist externally, such as when served as part of the same or linked HTTP connection. In addition, C2PA Manifest Stores may also be decoupled from assets, enabling the C2PA Manifest to be retrieved and processed separately from the asset.

2.5. Validation

Validation checks provenance claim signatures and performs supporting operations. The validation algorithm is described in detail in the validation section of the C2PA specification.

2.6. Protection of personal information

C2PA provides features that can be used to protect confidentiality of personal information while still establishing the provenance of an asset.

- Anonymous or pseudo-anonymous identities may be used for signing claims.
- Assertions that may contain sensitive information can be removed via redaction.
- Assertions can be customized for specific workflows, such as where specific information is required to remain encrypted end-to-end.
- Manifests may include W3C Verifiable Credentials.

Note that confidentiality of personal information and other privacy-related concerns are addressed in the Harms Considerations document.

Chapter 3. Security model

The C2PA security model is focused on security of C2PA provenance data. While attackers may aim to compromise the security of other aspects of the system, this model assumes the pivotal goal is compromise of C2PA provenance.

The following principles should hold:

Integrity

Attempts to compromise the integrity of C2PA provenance data associated with a respective asset will be detected upon validation. This includes all data stored within C2PA Manifests.

For embedded C2PA Manifests, malicious association of an existing manifest with a differing asset is also tamper-evident. In addition, the following actions are tamper-evident for all manifests:

- C2PA Manifest Store: Malicious addition or removal of a manifest.
- C2PA Manifest: Malicious modification of an active or ingredient manifest.
- Claims and claim signatures: Addition, removal, or modification of an existing claim in an existing C2PA Manifest.
- Assertions: Addition, removal, or modification of assertions in the assertion store referenced by an existing claim.

Availability

An attacker may degrade availability of C2PA data, but C2PA should provide means to mitigate this impact.

For example: An attacker may strip manifests from C2PA-enabled assets altogether. Guidance is provided on how to mitigate the impact of this scenario.

While C2PA supports cloud-hosted information that is referenced from assertions, the validation algorithm makes retrieval of that information optional for validation in order that any network attacks (e.g, DDOS, tracking, etc.) cannot impact user experience.

Confidentiality

C2PA does not specify confidentiality properties for assets, provenance model primitives, private address books, or cryptographic secrets.

- They may be kept confidential via other means. For example by transmitting assets via TLS. However, enforcing this confidentiality is considered out-of-scope for this specification.
- Some private data, such as claim signing keys or private address books, must adhere to conventional confidentiality principles. Compromise of this private data could lead to indirect violation of other principles specified in this section. Enforcing confidentiality of this adjacent private data is considered out-of-scope for this specification.
- C2PA does not specify Digital Rights Management (DRM) technology.

- C2PA does not specify guarantees of confidentiality of personal information.
    - However, C2PA provides features that can be used to protect confidentiality of personal information (summary)
    - Confidentiality of personal information and other privacy-related concerns are addressed in the Harms Considerations document.

      In general, these assertions are achieved through application of the C2PA Trust Model to the claim generation process, and corresponding checks in the validation process.

Non-repudiation

Manifests are signed with a digital signature to maintain the integrity of the data. This signature also provides the identity of the signing individual, organization, software system or hardware device. The content consumer will need to determine whether a certificate, and the associated certificate chain, is a trusted source.

To assist the content consumer, the validator is required to maintain a list of trust anchors to facilitate validation of the certificate chain. While the validator will trust the trust anchors in their list, a content consumer may need the ability to apply a additional criteria for trust for a given situation. Therefore, the content consumer will need to be able to view the trust anchors for the content. A validator may also allow the user to create and maintain a private credential store of signing credentials for each credential type. This store is intended as an "address book" of credentials they have chosen to trust based on an out-of-band relationship.

Some validators may be purpose-built for validating content from a finite set of sources who use a finite set of trust anchors. In these situations, the validator has the option to include only the trust anchors for those sources. This can reduce the risk of any conflicts or confusion arising from trust anchors outside of the finite set having certificates similar to those of the intended sources.

Chapter 4. Threat model

This section documents information security threats to C2PA technology. Threats are described as threat scenarios and organized categorically by prospective attacker goals; where an attacker goal is what the attacker is trying to achieve, or the desired outcome of the threat scenario.

Each threat scenarios is described as follows:

- Description: A short description of the threat scenario
- Prerequisites: Relevant requirements for attempting an attack or threat scenario
- Impact analysis: Summary of the impact of the threat scenario on the C2PA security model
- Guidance: Any supplementary security guidance on addressing the threat scenario or treating potential residual security risk

NOTE This document refers to threat sources, agents, actors and other malicious entities commonly as attackers.

NOTE For simplicity and readability, this document uses images as the example asset type for the threat scenarios. However, these threats would apply to all supported asset types within the C2PA specification.

4.1. Workflow diagram

Below is simple workflow diagram that represents a common C2PA workflow. Since each implementation may vary, this workflow diagram is not representative of any specific implementation. For instance, C2PA supports multiple asset types, and this diagram only represents the creation and viewing of a signed image. The intent of this diagram is solely to provide a high-level starting point for discussion.

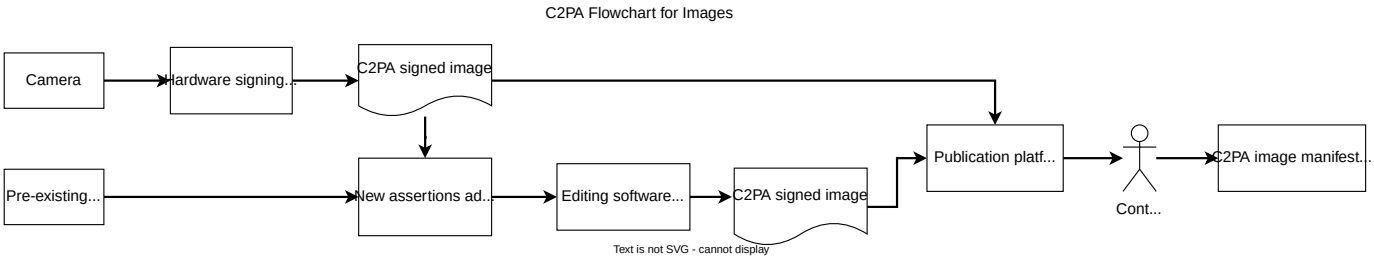

Figure 2. A C2PA workflow for images

4.2. Threat and attack assumptions

4.2.1. Assumptions

The following assumptions are made about entities that may attack C2PA technology:

- Attackers have full access to public assets and associated public metadata, including C2PA Manifests.
- Attackers may attempt to access claim generator software (e.g., C2PA-enabled photo editing software) as a typical user.
- Attackers do not have access to private keys referenced within the C2PA ecosystem (e.g., claim signing private keys, Time-stamping Authority private keys, etc.). They may, however, attempt to access these keys via exploitation techniques.
- Attackers have the ability to publish assets to an impactful audience, for example, as a social media user.
- Quantum computing cryptanalytic attacks are considered out-of-scope.

The following additional assumptions are made about the threat model:

- Attacks that rely on bypassing conventional security models of non-C2PA technology are considered out-of-scope. For example:
  - Attacks against the C2PA Validator as an architectural component within a user agent (e.g., web browser) are considered in-scope.
  - A compromise of the C2PA validation logic via some non-C2PA attack vector within the user agent is considered out-of-scope, since this would typically require bypassing the security model of the user agent (e.g., escaping a web browser sandbox or compromising the host via other means).
- Attacks across conventional trust boundaries are considered in-scope.
  - For example, attacks against C2PA-enabled assets traversing the internet.

4.2.2. Attack targets

4.2.2.1. Hardware layer:

Attacks against hardware implementations of C2PA would attempt to leverage the trust of the hardware manufacturer. For example, a camera lens sends digital information to the camera's CPU to sign what was captured. A hardware attacker might inject their own data between the lens and the camera's CPU to convince the camera to sign fake images.

4.2.2.2. Cryptographic PKI

Attacks against a PKI infrastructure might leverage several techniques. A simple example of this attack would be creating a self-signed certificate that spoofs a legitimate certificate in an attempt to exploit a flaw in the validator's implementation of trust list verification. Another example attack would be to socially engineer a certificate authority to sign content with another user's credentials. These attacks could also include attacking the cryptographic algorithms used in the signature. Lastly, the attacker could attempt to steal a signing key either through compromising a signing service, compromising a user's credentials to the signing service, or stealing the hardware used by the image creator. In addition to the guidance described below for specific threats, further cryptography recommendations are provided in the Trust section of the Guidance for Implementers.

4.2.2.3. User interface

Attackers targeting the user interface would create malicious manifest data that targets the user interface in which the user views the rendered content credentials. In this scenario, the attackers would embed malicious data into the C2PA metadata. When an end-user uses a user interface to view the C2PA metadata, the malicious data would be extraced from the asset and presented to the end user. This data could attempt to confuse the end-user directly. For instance, the attacker might try to use homoglyph attacks to confuse the end-user regarding the text that they are reading. Alternatively, the attacker might attempt to inject malicious data that could take control of the user interface and interfere with what is presented to the end-user. As an example, a flaw in a web-based user interface might allow the malicious manifest data to be interpreted as code by the web browser which would result in cross-site scripting. If the attacker's malicious payload is interpeted as JavaScript by the web browser, then the attacker could manipulate the trusted user interface into presenting false information to the end-user.

4.2.2.4. Consumers

Consumer attacks would focus on victimizing the human users of C2PA implementations. For instance, they might leverage how the C2PA specification allows remote manifests and references to other remote data to track and identify viewers of the content. When a person views Content Credentials, their client may make requests to these resources. If the remote resources are hosted by the attacker, the attacker could track which IP addresses have viewed specific assets. In countries where viewing certain types of content is prosecuted, this could result in the viewer's life being put at risk. In addition, it may be possible to frame someone for viewing banned content. An attacker could take a safe image and embed a banned image containing remote C2PA metadata as an ingredient. While the user would only see the safe image, they would be flagged for looking at banned content when the manifests for the embedded ingredient are resolved.

4.2.2.5. Application implementations

These attacks would focus on exploiting the software or hardware involved in C2PA Manifest processing. An example of this type of attack would be generating a C2PA Manifest with a sufficiently large payload to perform a buffer overflow. An attacker might also include malicious manifests that result in an image being misclassified by implementations that correlates C2PA information.

4.2.2.6. The C2PA specification

These attackers would have a thorough understanding in the C2PA specification and look for potential flaws in the C2PA guidance. While every effort is made to ensure that C2PA guidance is secure, a threat model would not be complete without taking into account the possibility for a mistake in the guidance or a change in the threat landscape that undermines the core tenets of the specification. Similar specifications have improved with each new version as a result of issues identified in the previous version. The threat model must take into account the possibility that the C2PA specification will have an unforeseen issue requiring an update to the specification and potential action by the C2PA community.

4.2.2.7. Community trust

One method for attacking the specification is to undermine people's trust in the technology. For instance, frequent

attacks spoofing information from C2PA Manifests, while not the direct fault of the specification, could wear on the community's patience and trust of the technology. If these attacks result in frequent efforts on the part of the end-user to verify that the content is not a spoofing attack, then they will lose patience from the extra required effort and abandon the technology. Once the community abandons the technology, attackers can revert to their original techniques for creating disinformation. The community needs be aware of malicious disruptors in the community and work to minimize their impact.

4.2.2.8. Misrepresentation and assumptions

These types of attacks are focused on human assumptions rather than technical attacks. For instance, one attack might be to take a photo of an image rendered by a high-definition television. The image captured by the device would be technically valid and realistic. However, without proper contextual information, a viewer of the image would assume that it was taken of a real event rather than a manufactured TV image. Similarly, there are also techniques for creating optical illusions for cameras. In these instances, the capture device correctly records and signs what came through the camera lens. However, the resulting image leads the viewer to a false conclusion about the event.

4.3. Threats

4.3.1. Goal: Degrading access to provenance information

Threat scenarios in this category aim to prevent media consumers from reasoning about the provenance of assets. In terms of the security model, this is a compromise of availability of C2PA provenance data.

4.3.1.1. Threat: Stripping C2PA Manifests

4.3.1.1.1. Description

An attacker removes C2PA Manifests from an asset delivered with hard bindings. For example, by downloading C2PA-enabled media from a social media site, stripping the C2PA Manifest(s), and re-posting it.

4.3.1.1.2. Prerequisites

Technical ability to strip provenance data from published or stored assets and deliver the stripped assets targeted users or systems. For example, via republishing to social media or on-path attacks.

4.3.1.1.3. Impact analysis

It is possible for an attacker to remove metadata as described. Depending on the context, this could cause a user agent not to display provenance information associated with the asset.

4.3.1.1.4. Security guidance

The impact of this threat scenario could be mitigated through use of manifest repositories, e.g., by making C2PA Manifests available to user agents when embedded manifests are missing.

Use of manifest repositories can introduce security and privacy issues. See the Content binding security section for more information.

4.3.1.2. Threat: Stripping data within C2PA Manifests

4.3.1.2.1. Description

An attacker removes metadata from within C2PA Manifests(s) stored in a asset delivered with hard bindings, such as the assertion store, individual assertions, claims, claim signatures, etc. For example, by downloading C2PA-enabled media from a social media site, stripping the target C2PA metadata, and re-posting it.

4.3.1.2.2. Prerequisites

Technical ability to strip provenance data from published or stored assets and deliver the stripped assets targeted users or systems. For example, via republishing to social media or on-path attacks.

4.3.1.2.3. Impact analysis

It is possible for an attacker to remove information from inside of a C2PA Manifest as described. However, validation checks as implemented by a C2PA validator will detect any complete or partial removal of data stored within C2PA Manifests due to the use of cryptographic hashes in the `hash-uri-map` structures stored within the claim. The data within the claim is protected by the claim signature. The `hashed-uri-map` structures within the Manifest Store are not a security protection which is why removal of an entire C2PA Manifest is possible, and described as a distinct threat scenario.

This is similar to the case of stripping all C2PA metadata. However, in this scenario there is an added benefit that user agents can explicitly detect the attempt to remove metadata as part of the validation process.

4.3.1.2.4. Security guidance

The impact of this threat scenario is mitigated through explicit detection of missing metadata through the standard C2PA validation process and factoring this into system behaviour. For example, a user agent would display the affected asset differently than a non-modified asset. (For more information about the user experience aspects, see C2PA's UX Guidance)

The impact of this threat scenario could be also mitigated through use of manifest repositories, e.g., by making decoupled C2PA Manifests available to user agents when embedded metadata is missing.

Use of manifest repositories can introduce security and privacy issues. See the Content binding security section for more information.

4.3.2. Goal: Tamper with provenance information

Threat scenarios in this category aim to modify or spoof provenance information associated with C2PA-enabled assets, e.g., to present misleading provenance data to media consumers. This is considered a compromise of integrity of provenance data in terms of the C2PA security model.

4.3.2.1. Threat: Copying C2PA metadata

4.3.2.1.1. Description

An attacker copies valid, hard-bound C2PA metadata, such as assertions, claims, or entire C2PA Manifests, from one C2PA-enabled asset to another.

4.3.2.1.2. Prerequisites

Technical ability to strip provenance data from published or stored assets and deliver the stripped assets targeted users or systems. For example, via republishing to social media or on-path attacks.

4.3.2.1.3. Impact analysis

An attacker is able to perform the copying as described. However, C2PA hard bindings provide a cryptographic binding between assets and C2PA Manifests via claim signatures. Thus, validation checks (e.g., as implemented by a user agent) will detect the replaced C2PA Manifests.

This is similar to the case of stripping all C2PA metadata. However, in this scenario there is an added benefit that user agents can explicitly detect the attempt to remove metadata as part of the validation process.

4.3.2.1.4. Security guidance

The impact of this threat scenario could be mitigated through explicit detection of incorrect metadata as described here, and factoring this into system behaviour. For example, a user agent would display the affected asset differently than a non-modified asset. (For more information about the user experience aspects, see C2PA's UX Guidance).

The impact of this threat scenario could be also mitigated through use of manifest repositories, e.g., by making decoupled C2PA Manifests available to user agents when embedded metadata is missing.

Use of manifest repositories can introduce security and privacy issues. See the Content binding security section for more information.

4.3.2.2. Threat: Spoofing signed C2PA metadata via stolen key

4.3.2.2.1. Description

An attacker compromises the confidentiality of a claim signing key and uses it to attach valid, malicious provenance to a asset.

4.3.2.2.2. Prerequisites

Ability to read and use a valid C2PA claim signing key. For example by stealing it from a claim generation and signing system. Ability to copy, modify, or generate assets, add desired (valid) C2PA provenance metadata to them, and deliver them to targeted users or systems. For example, via republishing to social media or on-path attacks.

4.3.2.2.3. Impact analysis

An attacker that is able to steal a signing key can spoof valid C2PA Manifests on arbitrary assets. This applies to both hard and soft bindings. Valid claim signatures will be limited to the exposed key; an attacker cannot impersonate other signers in this scenario. These manifests will pass validation checks. Further, an attacker will be able to continue to generate assets with spoofed provenance until the compromised credential is revoked, and revocation status is propagated to the targeted users/systems.

If the signer's credential type does not support or provide a means for revocation, it may be necessary to remove the trust anchor associated with the signer's credential from the respective C2PA trust list to effectively revoke the malicious manifests. In this case, all manifests associated with the trust anchor will be invalidated.

Revocation of the signing key associated with the malicious manifests may invalidate other manifests if signing keys are reused between claim generation / claim signing system users.

4.3.2.2.4. Security guidance

Claim generators must adhere to best practices for securing keys in their respective operating environments. This includes using platform-specific best practices for key storage and management (e.g., hardware security modules, key management services), minimizing key reuse, and revoking keys when compromise is suspected. For more information on key management, see the NIST Key Management Guidelines.

It is highly recommended that all signing credentials include revocation information.

4.3.2.3. Threat: Spoofing signed C2PA metadata via misuse of claim generator

4.3.2.3.1. Description

An attacker misuses a legitimate claim generator (e.g., photo editing service) to add misleading provenance to a C2PA-enabled asset.

4.3.2.3.2. Prerequisites

Ability to access and exploit or misuse a claim generator system. For example, misuse of a claim generator may result in malicious additions of C2PA Manifests or Claims to assets; exploitation of vulnerabilities in claim generators may lead to modification or tampering with existing C2PA Manifest data added with the target claim generator's signing key(s). Ability to deliver assets to targeted users. For example, via publishing to social media or on-path attacks.

4.3.2.3.3. Impact analysis

An attacker may be able to misuse or exploit a claim signing system to create malicious C2PA Manifests. These can then be delivered to targeted users/systems. Note that valid claim signatures will be limited to credentials used by the targeted claim signing system; an attacker cannot impersonate other signers in this scenario.

If the signer's credential type does not support or provide a means for revocation, it may be necessary to remove the trust anchor associated with the signer's credential from the respective C2PA trust list to effectively revoke the malicious manifests. In this case, all manifests associated with the trust anchor will be invalidated.

Revocation of the signing key associated with the malicious manifests may invalidate other manifests if signing keys are reused between claim generation / claim signing system users.

4.3.2.3.4. Security guidance

Claim generators must adhere to industry best practices for information security and anti-abuse practices, including leveraging available platform-specific features for deployment (e.g. Android SafetyNet, Apple DeviceCheck). Likewise, claim generators should use platform-specific best practices for key storage and management. For example, usage of hardware security modules, key management services), minimizing key reuse, and revoking keys when compromise is suspected. For more information on key management, see the NIST Key Management Guidelines.

It is highly recommended that all signing credentials include revocation information.

4.3.2.4. Threat: Issuance of claim signing keys to malicious entity

4.3.2.4.1. Description

An attacker causes a C2PA claim signing key issuer to issue a valid signing key to them. They then use this signing key to add misleading provenance to a C2PA-enabled media asset.

4.3.2.4.2. Prerequisites

Ability to bypass unspecified processes for obtaining a claim signing key.

4.3.2.4.3. Impact analysis

An attacker that is able to obtain a legitimate claim signing key can spoof valid C2PA Manifests on arbitrary media assets. These manifests will pass validation checks. Further, an attacker will be able to continue to generate media assets with spoofed provenance until the compromised credential is revoked, and revocation status is propagated to the targeted users/systems.

It may be necessary to remove the trust anchor associated with the signer's credential from the respective C2PA trust list to effectively revoke malicious manifests. In this case, all manifests associated with the trust anchor will be invalidated.

4.3.2.4.4. Security guidance

The C2PA does not currently specify requirements regarding issuance of claim signing keys. It is recommended that these processes be robust against exploitation and misuse. The CAB Forum Baseline Requirements provide a working example of security requirements for signing credential issuers.

4.3.2.5. Threat: Swapping ingredients

4.3.2.5.1. Description

An attacker finds a media asset with a C2PA Manifest Store that includes an ingredient manifest with a signature from valid, non-public, credentials. They replace the ingredient manifest with a malicious manifest.

4.3.2.5.2. Prerequisites

Ability to discover media with susceptible ingredient manifest and generate a malicious replacement. And deliver the media asset to the target users/systems.

4.3.2.5.3. Impact analysis

Validation of ingredients is optional, however, validation of the hash of the ingredient's manifest at the time it was added to the active manifest as an ingredient is not. See the Validation section of the C2PA specification for details.

While a C2PA validator is not expected to have access to the original, non-public, signing credential nor to the original ingredient asset, it can differentiate between the original and compromised provenance by checking the `hashed_uri` present in the ingredient assertion. Since the malicious manifest will not match the original, the validator is able to detect the attack.

4.3.2.5.4. Security guidance

Claim generators should add review ratings to their assertion metadata to propagate their assessment of ingredients to downstream consumers. When this validation is performed on both media assets described in this scenario, downstream consumers will be able to differentiate between them.

4.3.2.6. Threat: Binding malicious media asset to valid provenance via soft-binding collision

4.3.2.6.1. Description

An attacker generates a malicious media asset whose soft binding matches the soft binding of a legitimate, target media asset. The attacker creates a decoupled, soft-bound manifest for the malicious media asset and injects it into a manifest repository. The malicious media asset is then delivered to the target/system user, which retrieves the legitimate C2PA Manifest via the common soft binding.

4.3.2.6.2. Prerequisites

Ability to generate a colliding soft binding for a target media asset and publish the resulting manifest such that it will be retrieved by the target user/system for the target media asset. Also requires that access to be able to add manifests to a manifest repository.

4.3.2.6.3. Impact analysis

Malicious media bound to legitimate C2PA provenance data is validated and presented to the target system/user.

4.3.2.6.4. Security guidance

C2PA provides guidance on applying soft bindings, which may include algorithms for mitigating this type of attack. See the content binding documentation for details.

4.3.2.7. Threat: Use valid W3C Verified Credentials to add authenticity to incorrect actor attribution

4.3.2.7.1. Description

An attacker copies a valid W3C Verified Credential (VC) from an existing C2PA Manifest. The attacker then creates a separate, valid C2PA Manifest that includes the copied VC, misleading information, and a valid claim signature. The resulting manifest is presented to consumers.

4.3.2.7.2. Prerequisites

- Ability to copy a valid VC from an existing C2PA Manifest
- Ability to generate and sign a claim that passes validation
- Ability to deliver assets to consumer(s), for example via publishing to social media

4.3.2.7.3. Impact analysis

Consumers are presented with UX that indicates the subject of copied VC is associated with the manipulated asset.

4.3.2.7.4. Security guidance

User agents must make a clear separation between the signer of the Claim and any included W3C Verifiable Credentials. It must be clear that the association of those VCs to assertions, are made by the claim signer (i.e., the subject of the VC was not responsible for the association).

Claim generators and signing operations should adhere to security best practices.

4.3.2.8. Threat: Binding valid media asset to malicious provenance via soft-binding collision

4.3.2.8.1. Description

An attacker generates a dummy media asset whose soft binding matches the soft binding of another media asset. They then use a claim generator to create a valid, malicious C2PA claim(s), and a C2PA Manifest. They then publish the C2PA Manifest to a manifest repository. When the target media asset is validated, the malicious C2PA Manifest is retrieved via the common soft binding.

4.3.2.8.2. Prerequisites

Ability to generate a colliding soft binding for a target media asset and publish the resulting manifest such that it will be retrieved by the target user/system for the target media asset.

4.3.2.8.3. Impact analysis

Malicious provenance data bound to a legitimate media asset is validated and presented to the target system/user.

4.3.2.8.4. Security guidance

C2PA provides guidance on applying soft bindings, which may include algorithms for mitigating this type of attack.

See the content binding documentation for details.

4.3.2.9. Threat: Recovering redacted information

4.3.2.9.1. Description

Each manifest has a digital signature that includes a hash of the original content. An assertion is redacted by using zeroes to overwrite the relevant section of the manifest. However, the original hash remains a part of the image via the signature. The redaction information includes information on the type of data that was redacted from the original manifest. An attacker may try to guess what was originally contained within the assertion by brute-forcing the potential values until they find a value that matches the original hash. While many bytes will have been replaced with zeroes, the standardization of the formatting means that many of those bytes will be predictable. In addition, brute forcing of all possible values may not be necessary for certain situations. For instance, an attacker may just want to confirm whether the image was created using a short list of individuals that they are interested in. Therefore, they would only need to test whether author assertions with those values match the hash. Some data types, such as geolocation, have a finite set of overall values and even smaller set of likely values if you can make certain assumptions about where the photo might have been taken. Therefore, without mitigations, it might be possible to brute force the original values of certain redacted data types.

4.3.2.9.2. Prerequisites

The attacker needs to have access to a compute resource capable of brute-forcing the expected space of the redacted field that they are targeting. If a unique, cryptographically random C2PA salt was included with the assertion, then brute-forcing the original value should not be feasible.

4.3.2.9.3. Impact analysis

The ability to recover redacted data would jeopardize any safety and security provided by the redaction.

4.3.2.9.4. Security guidance

A C2PA salt must be included with each assertion. The salt must be cryptographically random, unique for each assertion, and either 16 or 32 bytes in length. This approach results in the attacker having to brute force both the combination of original value and the randomized salt corresponding to that value. The inclusion of randomized salt should make this technique infeasible.

4.3.3. Goal: Exploit functionality introduced by C2PA

Threat scenarios in this category do not aim to break the C2PA security model. Rather, they aim to leverage C2PA functionality to perform systems attacks or exploitation. Note that the set of scenarios included here is not meant to be comprehensive; they focus on application of known exploitation techniques against behaviour explicitly referenced in the C2PA specification. It should be noted that the C2PA specification does not provide any requirements to validate what exists inside of an assertion. Therefore, applications must verify that the data within an assertion is not a risk to their implementations.

4.3.3.1. Threat: Interfering with UI rendering of C2PA metadata

4.3.3.1.1. Description

A common attack for this category would be cross-site scripting. An attacker could inject malicious JavaScript into a claim. When a web user interface renders the claim details for an end-user, the malicious JavaScript within the claim may get executed by the web browser. This could result in manipulating the behaviour of the trusted user interface and misleading the end-user. Another attack in this category might be the injection of special characters that prevent the UI from displaying the full data included in the claim such as nulls or carriage returns.

4.3.3.1.2. Prerequisites

The attacker needs to have access to a claim signing mechanism that will allow for arbitrary text within signed claims. This could be a self-signing mechanism if the targeted implementation does not limit content based on the certificate authority.

4.3.3.1.3. Impact analysis

The viewer of the claim metadata will be misled with regards to the validity of the image. As an example, if the malicious payload contained attacker controlled JavaScript that was interpreted by the web page displaying the C2PA metadata to the end-user, then the attacker could manipulate the UI so that the viewer would see fake signing information or incomplete claim information.

4.3.3.1.4. Security guidance

Prior to claim signing, signers can verify that the fields provided contain the expected characters, length, and format for the given field. When displaying Content Credentials, implementations can leverage output encoding to prevent special characters from being interpreted as code.

4.3.3.2. Threat: Interference with the storage and organization of C2PA manifest data

4.3.3.2.1. Description

Many organizations may try to index and store C2PA data for the purposes of search, data analytics, or similar functionality. An attacker may want to prioritize their image in the result by interfering with the classification of this image. A likely attack for this scenario would be SQL injection in order to manipulate the database of stored information.

4.3.3.2.2. Prerequisites

The attacker needs to have access to a claim signing mechanism that will allow for arbitrary text within signed claims. This could be a self-signing mechanism if the targeted implementation does not limit content based on the certificate authority.

4.3.3.2.3. Impact analysis

The implementation attempting to store and organize the data may become controlled by the attacker.

4.3.3.2.4. Security guidance

Implementations storing data within a database should follow the security best practices for their database solution. As an example, common best practices for SQL database include prepared statements and stored procedures. The implementation may also choose to implement data validation prior to storing the metadata and reject any images that have potentially malicious data.

4.3.3.3. Threat: Exploitation of the application software

4.3.3.3.1. Description

Applications written in memory unsafe languages, such as C and C++, may be vulnerable to techniques such as buffer overflow, heap overflow, and use-after-free attacks. All applications, regardless of language, need to maintain any third-party libraries in use to ensure that they are fully patched. In addition, all applications may be vulnerable to flaws in the application logic resulting in attacker-controlled conditions.

4.3.3.3.2. Prerequisites

The attacker needs to have access to a claim signing mechanism that will allow for arbitrary text or the ability to manipulate the binary structures within signed claims. This could be a self-signing mechanism if the targeted implementation does not limit content based on the certificate authority. For web-based applications, this could also include the ability for the attacker to control the parameters sent within a web request.

4.3.3.3.3. Impact analysis

These types of attacks frequently lead to system compromise of the environment in which the application is running, such as an end-user's desktop. If the attacker controls the system, then the system can no longer be trusted for any function.

4.3.3.3.4. Security guidance

The implementation should consider using memory safe languages for its implementation. Data validation should be performed to ensure that inputs are of the proper length and format. An implementation may also choose to perform fuzzing to validate the software can robustly handle unexpected input. Have a process to ensure that third-party dependencies are kept up to date. Perform regular security testing of the software to identify any potential logic flaws in the implementation. If possible, conduct active monitoring of the environment to detect attack attempts. Additional information can be found in the Validation section of the Guidance for Implementers .

4.3.3.4. Threat: Exploitation of the hosting environment

4.3.3.4.1. Description

An attacker may choose to target the infrastructure of the C2PA implementation. These attacks could include exploitation of misconfigurations within the hosting environment, exploitation of hosting provider, or network security vulnerabilities.

4.3.3.4.2. Prerequisites

The attacker needs to be able to identify a flaw in the implementation's services or configuration that can be remotely exploited.

4.3.3.4.3. Impact analysis

The attacker could gain control of the hosting environment for the C2PA implementation making the C2PA implementation untrustworthy.

4.3.3.4.4. Security guidance

Implementations should ensure that they follow all hosting provider's security best practices. Server software and hardware should be kept up to date. Conduct regular network penetration testing against the environment to ensure that no known vulnerabilities are present. In addition, ensure that there is active monitoring of the environment to detect any attacks.

4.3.4. Goal: Confusing or misleading a human interpretting C2PA information

Threat scenarios in this category aim to confuse the human consumer trying to evaluate the information being presented to them. These attacks do not violate the technical requirements of the C2PA specification. However, a malicious entity is trying to present information that misleads, confuses, or otherwise tries to establish a trust with the consumer that the consumer wouldn't normally want to provide. For instance, this category would include impersonation attacks where an attacker is trying to impersonate a trusted entity.

4.3.4.1. Threat: Name collision attacks

4.3.4.1.1. Description

An attacker tries to leverage the trust in a known entity by obtaining a certificate with the same, or similar, name. One method for this attack is for the attacker using a different certificate authority than the person that they are trying to impersonate. For instance, assume that example.org's identitiy is provided by Alice's Trusted Certificate Authority. An attacker may try to request an example.org certificate from Bob's Trusted Certificate Authority. Since Alice and Bob are two different groups, the collision goes unnoticed. The consumer of the C2PA manifest may be unaware of whether example.org uses Alice or Bob's trusted CA for their certificates. Name collisions can also occur without malicious intent when dealing with human identity since many people have the same name.

4.3.4.1.2. Prerequisites

The attacker needs to have the ability to request a certificate from a trusted certificate authority using the identity of the target.

4.3.4.1.3. Impact analysis

The viewer of the claim metadata will be misled with regards to the author of the C2PA content. This can result in the viewer mistakenly assigning trust to something that they would consider untrusted.

4.3.4.1.4. Security guidance

When providing identity information to manifest consumers, provide the details of the full certificate chain. If possible, leverage the certificate hash or other available unique identifiers to verify that the certificate represents the expected identity. If possible, leverage certificate transparency logs to monitor for potential certificate collisions.

4.3.4.2. Threat: Assumed trust relationship between two identies

4.3.4.2.1. Description

C2PA content can have multiple identities associated with it. However, those identities may not have a trust relationship. Assume that the base of the content is signed by a major, trusted news organization. Changes are then made to the content and those changes are signed by a malicious individual. The consumer of the content may assume that the malicious individual is associated with the trusted news organization, and therefore trust the content based on the news organization's association with the content. The certificate used by the malicious individual may or may not include a common name that tries to assist in confusing the user regarding their relationship to the news organization.

4.3.4.2.2. Prerequisites

The attacker needs to have a valid certificate that is included in a trust list used by the platform presenting the C2PA metadata.

4.3.4.2.3. Impact analysis

The viewer of the claim metadata will be misled with regards to the association between the two groups and assign trust based on the more recognizable entity.

4.3.4.2.4. Security guidance

When architecting your organization's signing workflow, consider making the brand signature the final signature over the entire content. If possible, ensure that the certificates used by employees of an organization share the same certificate authority. Another approach is to create an assertion that links the two identities. A user interface needs to clearly present all identities present within C2PA content, and the order in which those identities modified the content.

Chapter 5. Additional Implementation Best Practices

This section includes best practice guidance for implementors of the C2PA specification. Every implementation is unique and therefore this information is not all-inclusive. This section is intended to provide best practices that are recommended to help ensure the security of the C2PA community.

5.1. Tracking, transparency, and privacy

Customers should have a clear understanding about how C2PA implementations function and their supported use cases. The C2PA specification can support a wide range of use cases ranging from anonymous journalism to basic attribution for creative artists. More complex use cases may require additional protections outside the scope of this specification. C2PA implementors should be transparent with customers about the use cases that they intend to support.

5.1.1. Anonymity

5.1.1.1. Anonymity of the C2PA author

One potential use case for C2PA is anonymous journalists who are at risk of physical harm if they are caught. Supporting this use case would involve protections outside of what is detailed in the C2PA specification such as legal response plans and specialized logging. If the solution is not designed to support the anonymous journalist use case, then those types of users should be able to easily discern that the solution is not appropriate for their needs.

5.1.1.2. Opting into or out of C2PA metadata inclusion

Customers will need to know how to control the generation of C2PA metadata in their content. Accidental or unintentional inclusion of metadata could cause harm to at risk groups. For instance, children are often a specialized use case due to additional legal protections and potential threats. In addition, there may be abuse victims who need to control their ability to be tracked by those who could cause them harm. Providing transparency about how to disable C2PA tracking can help protect these users.

5.1.1.3. Anonymity of the content viewer

Remote C2PA content credential references could be used to track viewers of a piece of content against their will. A malicious author might intentionally use remote references to track anyone who views the image to identify their perceived enemy. One protection option is to warn the user that the content has remote credentials and obtain confirmation as to whether they want that information retrieved. If consent is granted, consider proxying the fetching of that information for the end-user. This action would prevent the end-user's IP address from being disclosed to the remote site. Regardless of implementation, it is encouraged to inform content consumers how data will be fetched.

5.1.2. Separation of identities

Content authors may need to be able to support multiple profiles for different use cases. Freelance artists and

journalists may work for more than one employer and may need to split their identities to reflect their different employment situations. People may also want to split their professional and personal identities. Lastly, certain types of artists may want to publish using a pseudonym, such as a nom de plume or stage name. Therefore, provide clarity to end-users about how to best control their identities and the identity of their devices.

5.2. Incident response

It is likely that highly skilled and highly resourced adversaries will be targeting C2PA infrastructure. Therefore, C2PA implementations should have incident response plans for when they are targeted. Incident response plans should be prepared for all aspects of the C2PA implementation and ecosystem. A breach could occur in the implementation's software or hosting environment. There could be a breach of an end-user, such as an account compromise. In addition, there could be a breach of C2PA dependency, such as a certificate authority. Practice executing the incident response plans on a regular basis to ensure that any attack can be quickly mitigated with minimal system impact.