Network Working Group
Internet-Draft
Intended status: Experimental
Expires: January 6, 2016

D. Zhang

D. Gillmor
CMRG
D. He
Huawei
B. Sarikaya
Huawei USA

July 5, 2015

CT for Binary Codes
draft-zhang-trans-ct-binary-codes-03

Abstract

This document proposes a solution to use Certificate Transparency for publishing software (or the digest of binary codes) and their signatures.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Digital signatures have been widely used in software distributions to demonstrate the authenticity of software. Through verifying signatures, an end user can ensure that the software she gets is developed by a legal provider (e.g., Microsoft) and is not tampered during the distribution. If an end user does not have a direct trust relationship with the software provider, an authentication chain needs to be generated from the key used for generating the signature to a trust anchor that the user trusts. That is why many signature mechanisms for software distribution are based on public key infrastructure (PKI).

However, signatures cannot prevent software provider from distributing software with customized backdoors/drawbacks. In some circumstances, it may be hard for a user to detect the differences between the software it got and the software provided to other users.

This draft describes a mechanism which extends the Certificate Transparency (CT) specified in [I-D.ietf-trans-rfc6962-bis] to support logging binary codes. A software provider can publish its binary codes (or digests of codes in order to e.g., save space or

avoid violating license restrictions) to one or more CT logs. Therefore, a user can easily detect whether there are customized backdoors by monitoring the log records.

IIn this mechanism, after a section of binary codes has been published to a log, the log will return a ticket (called Signed Certificate Timestamp (SCT) in this case) to the software provider. In this mechanism, the software without the ticket MUST NOT be trusted even if it is associated with a proper signature. This approach then forces the software providers to publish their binary codes to logs before distributing them.

2. Cryptographic Components of Certificate Transparency

The introduction of cryptographic components of CT is in Section 2 of [I-D.ietf-trans-rfc6962-bis]. When applying CT for binary codes, a log is a single, ever-growing, append-only Merkle Tree of Delegation Signer (DS) Resource Records (RRs).

3. Motivation Scenarios

The documents disclosed by Edward Snowden have raised the concerns of people on the vulnerability of the network devices to the passive attacks performed by NSA or other organizations. Some vendors also meet problems in their foreign markets because their products are suspected to have customized backdoors for adversaries to perform attacks. It is desired for vendors to publish the design details of the products and provide sufficient facilities for clients to check whether certain hardware or software of a device has been improperly modified. There are various techniques that could be used for this purpose. One way is to force a vendor to publish the binary codes of its firmwares. Therefore, customers can easily detect whether the vendor is releasing the same firmware to everyone. In addition, under the assistance of the binary transparency, customer will have more confidence on the quality of firmware. Since the same codes are used by different customers all over the world, the drawbacks in firmware will be easier to be detected.

There are similar requirements to detect the customized backdoors in the software market. Besides the software itself, a user may also concern whether there are customized backdoors in the patches. the binary transparency can help address such concerns in the same way. In addtion, this mechanism can also show some advantages in the scenarios where the signer does not realize that their keys have been compromised. If their update system requires using a CT log they could find out about their compromise.

4. Log Format and Operation

4.1. Log Entries

A developer/issuer of software can submit the software and the associated signature to any preferred CT logs before distributing it. In some cases, the software developer may select only to publish the signed digest of the software because of the license restriction or the space restriction of log record. In order to verify the attribution of each log record, a log SHALL publish a set of certificates that it trusts to benefit an issuer to construct an authentication chain connecting a trust anchor and the certificate containing the key used to sign the software.

A log needs to verify the authentication chain provided by the issuer, and MUST refuse to accept the signed software/digest if the chain cannot lead back to a trusted anchor. If the software/digest and the signature are accepted by a log and an SCT is issued, the log MUST store the entire chain and MUST present this chain for auditing upon request.

To comply with the certificate entries specified in [I-D.ietf-trans-rfc6962-bis], each software entry in a log MUST include the following components:

```
    enum { x509_entry(0), precert_entry(1), BIN_entry(TBD1), (65535) }
LogEntryType;
    enum { binary(TBD3), binary_digest(TBD4) } Signed_Type;

    struct {
        LogEntryType entry_type;
        Signed_Type signed_type;
        select (entry_type) {
            case x509_entry: X509ChainEntry;
            case precert_entry: PrecertChainEntry;
            case BINARY_entry:SigSoft_Chain_Entry
        } entry;
    } LogEntry;

    opaque BINARY<1..2^24-1>;
    struct {
       BINARY signed_software;
      ASN.1Cert certificate_chain<0..2^24-1>;
    } BINARY_Chain_Entry;
```

"entry_type" is the type of this entry. the type value of a binary log entry is TBD1.

signed_type indicates whether the signature is generated based on the software or its digest.

"signed_software" consists a ContentInfo structure specified in CMS[RFC5652]. Specifically, this field includes the binary codes/ digest, the signature, and any other additional information used to describe the software and the issuer publishing the software. The software SHOULD encapsulated and signed following the ways specified in CMS[RFC5652] . If signed_type is TBD3, the software is encapsulated in this field. If signed_type is TBD4, the SHA-256 digest of software is encapsulated in this field.

"certificate_chain" includes the certificates constructing a chain from the certificate of issuer to a certificate trusted by the log. The first certificate MUST be the certificate of issuer. Each following certificate MUST directly certify the one preceding it. The final certificate MUST either be, or be issued by, a root certificate accepted by the log. if the informaiton chain is provided in the signed_software field, this field is set to empty.

4.2. Structure of the Signed Certificate Timestamp

This work reuses the structure of Signed Certificate Timestamp (SCT) specified in Section 3.3 of [I-D.ietf-trans-rfc6962-bis] but makes necessary extensions.

```
    enum { certificate_timestamp(0), tree_hash(1),BINARY_timestamp(TBD2),
(255) }
      SignatureType;

    enum { v1(0), (255) }
      Version;

      struct {
          opaque key_id[32];
      } LogID;

     opaque digestcodes<0..2^24-1>;
     struct {
        opaque issuer_key_hash[32];
        digestcodes binary_digest;
      } Binary_Codes;

      opaque CtExtensions<0..2^16-1>;
```

"key_id" and "issuer_key_hash" are defined in Section 3.3 of [I-D.ietf-trans-rfc6962-bis].

binary_digest is the SHA-256 hash of binary codes.

```
       struct {
           Version sct_version;
           LogID id;
           uint64 timestamp;
           CtExtensions extensions;
           digitally-signed struct {
               Version sct_version;
               SignatureType signature_type = DSRR_timestamp;
               uint64 timestamp;
               LogEntryType entry_type;
               select(entry_type) {
                   case x509_entry: ASN.1Cert;
                   case precert_entry: PreCert;
                   case BINARY_entry: Binary_Codes;
               } signed_entry;
              CtExtensions extensions;
           };
       } SignedCertificateTimestamp;
```

"sct_version", "timestamp", "entry_type" are are identical to what is defined in Section 3.3 of [I-D.ietf-trans-rfc6962-bis].

5. Log Client Messages

In Section 4 of [I-D.ietf-trans-rfc6962-bis], a set of messages is defined for clients to query and verfiy the correctness of the log entries they are interested in. In this memo, two new messages are defined for CT to support binary transparency.

5.1. Add Binary and Certificate Chain to Log

POST https://<log server>/ct/v1/add-Binary-chain

Inputs:

software: the binary code(or digest), the signature, and the information used to describe the software and the signer publishing the software, which are encapsulated following the ways specified in CMS[RFC5652] .

chain: An array of base64-encoded certificates. The first element is the certificate used to sign the binary codes; the second chains to the first and so on to the last, which is either the root certificate or a certificate that chains to a

known root certificate. If the certificate chain information has been incuded in the software field, this field could be empty.

Outputs:

sct_version: The version of the SignedCertificateTimestamp structure, in decimal. A compliant v1 implementation MUST NOT expect this to be 0 (i.e., v1).

id: The log ID, base64 encoded.

timestamp: The SCT timestamp, which is the current NTP Time [RFC5905], measured since the epoch (January 1, 1970, 00:00), ignoring leap seconds, in milliseconds.

extensions: An opaque type for future expansion. It is likely that not all participants will need to understand data in this field. Logs should set this to the empty string. Clients should decode the base64-encoded data and include it in the SCT.

signature: The SCT signature, base64 encoded.

6. IANA Considerations

This document specified a new LogEntryType value TBD1, and a new SignatureType value TBD2.

7. Security Considerations

8. Acknowledgements

9. Normative References

[I-D.ietf-trans-rfc6962-bis]
Laurie, B., Langley, A., Kasper, E., Messeri, E., and R. Stradling, "Certificate Transparency", draft-ietf-trans-rfc6962-bis-07 (work in progress), March 2015.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, September 2009.

[RFC5905] Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.

Authors' Addresses

Dacheng Zhang

Email: dacheng.zhang@gmail.com

Daniel Kahn Gillmor
CMRG

Email: dkg@fifthhorseman.net

Danping He
Huawei

Email: ana.hedanping@huawei.com

Behcet Sarikaya
Huawei USA
5340 Legacy Dr. Building 3
Plano, TX 75024

Email: sarikaya@ieee.org