Network Working Group
Internet-Draft
Intended status: Experimental
Expires: April 30, 2015

D. Zhang
Huawei

October 27, 2014

CT for Binary Codes
draft-zhang-trans-ct-binary-codes-00

Abstract

This document proposes a solution to use CT for publishing softwares/ binary codes and their signatuers.

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

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

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

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

This Internet-Draft will expire on April 30, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

Digital signatures have been widely used in software distributions to demonstrate the authenticity of softwares. Through verifying signatures, end users can could ensure that the softwares they got are provided by legal organizations and never modified during the delivery. If an end user does not have a direct trust relationship with the software provider, an authentication chain need 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 PKI.

However, signatures cannot prevent a software developer from distributing softwares 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 describe a mechanism to extend Certificate Transparency specified in [I-D.ietf-trans-rfc6962-bis] to support logging binary codes. A software provider can publish its softwares (or digests of binary codes in the cases the softwares are non-trival) to one or more logs. Therefore, a user can easily detect the customized backdoors through monitor the log entries.

In this mechanism, after a section of binary codes is published to a log, the log will return a ticket (called Signed Certificate

Timestamp (SCT) in this case) to the software provider. The software without the ticket will not be trusted even it is associated with a proper signature. This approach will force providers to publish their binary codes to logs.

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 DS RRs.

3. Motivation Scenarios

3.1. Dealing with Customized Backdoors in Firmwares

The disclosed documents 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 have met problems in the abroad 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 functions 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, the binary transparency, customer will have more confidence on the quality of firmwares. Since the same codes are used by different customers all over the world, the drawbacks in firmwares will be easier to be detected.

3.2. Another Scenarios

Ben suggested to us CT for publishing FreeBSD. Can you contribute some ideas here?

4. Log Format and Operation

4.1. Log Entries

A developer of a sfotware can submit the signed software (or the digest of the software to save space) to any preferred logs before distributing it. In order to enable attribution of each logged RR to its issuer, the log SHALL publish a list of certificates to construct an authentication chain connecting the trust anchor and the certfiicate used to sign the software.

Logs MUST verify that the authentication chain and make sure it leads back to a trusted certificate, using the chain of certificates provided by the submitter. Logs MUST refuse to publish a signed software without a valid chain. If the software and the signature are accepted and an SCT issued, the accepting log MUST store the entire chain used for verification, including the signed software itself and including the trusted zone signing key used to verify the 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;

 struct {
     LogEntryType entry_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_software" include the binary codes, the signature, and any other additional information used to describe the software and the signer publishing the software. The current version of the document does not specify how to structure such information. (CMS[RFC5652] can be used.) This work will be left for future work.

"certificate_chain" include the certificates constructing a chain from the certificate of signer to a certificate trusted by the log.The first certificate MUST be the certificate of signer. 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.

4.2. Structure of the Signed Certificate Timestamp

This work reuses the structure of Signed Certificate Timestamp specified in Section 3.3 of [I-D.ietf-trans-rfc6962-bis] but make 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. (Open question: In the future version of the this document, do we need to support other digest algorithms? If so, maybwe we should provie an digest identifier field here.)

```
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 and extensions" are are identical to what is defined in Section 3.3 of [I-D.ietf-trans-rfc6962-bis]..

"extensions" are future extensions to this protocol version (v1). Currently, no extensions are specified.

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, the signature, and the information used to describe the software and the signer publishing the software

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.

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, in decimal.

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 makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

7. Security Considerations

8. Acknowledgements

9. Normative References

[I-D.ietf-trans-rfc6962-bis] Laurie, B., Langley, A., Kasper, E., and R. Stradling, "Certificate Transparency", draft-ietf-trans-rfc6962-bis-04 (work in progress), July 2014.

[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.

Author's Address

Dacheng Zhang
Huawei

Email: zhangdacheng@huawei.com