CT for Binary Codes

draft-zhang-trans-ct-binary-codes-04

Liang Xia Huawei
DaCheng Zhang Huawei
Daniel Kahn Gillmor CMRG
Behcet Sarikaya Huawei

March 2017 Chicago
Key Goal

• Transparently logging the software binary codes (BC) or its digest with their signature to:
  – enable anyone to monitor and audit the software provider activity: misdistribution by illegal software provider
  – notice the distribution of suspect software: tampered software with customized backdoors/drawbacks
  – audit the BC logs themselves: inconsistency of software among different BC logs
How to Realize

• **Extending the Certificate Transparency protocol** [I-D.ietf-tf-trans-rfc6962-bis]:
  – **Logging** what: software binary codes (BC) or its digest with their signature vs TLS server certificates
  – **Issuing** what: Signed Binary Timestamp vs Signed Certificate Timestamp
  – **Log Format Extension**: new TransItem, new Merkle Tree leaves definition, new SBT definition;
  – **Log Client Messages’ change**: Add log, Retrieve Entries and STH from Log
  – Others: remain the same
New Binary Transparency Log Entries

enum {binary(TBD1), binary_digest(TBD2)} BIN_Signed_Type;

opaque BINARY<1..2^24-1>;
opaque ASN.1Cert<1..2^24-1>;
struct {
    BIN_Signed_Type bin_signed_type;
    BINARY signed_software; //binary code/SHA-256 digest of software, signature, other, CMS[RFC5652]
    ASN.1Cert certificate_chain<1..2^24-1>;
} BinaryChainEntryV2;
Extensive TransItem Structure

definitions:

```c
enum {
    reserved(0),
    x509_entry_v2(1), precert_entry_v2(2),
    x509_sct_v2(3),
    precert_sct_v2(4),
    signed_tree_head_v2(5), consistency_proof_v2(6),
    inclusion_proof_v2(7), x509_sct_with_proof_v2(8),
    precert_sct_with_proof_v2(9), BIN_entry_v2(TBD3),
    BIN_sbt_v2(TBD4), BIN_sbt_with_proof_v2(TBD5),
    (65535)
} VersionedTransType;

struct {
    VersionedTransType versioned_type;
    select (versioned_type) {
        case x509_entry_v2: TimestampedCertificateEntryDataV2;
        case precert_entry_v2: TimestampedCertificateEntryDataV2;
        case x509_sct_v2: SignedCertificateTimestampDataV2;
        case precert_sct_v2: SignedCertificateTimestampDataV2;
        case signed_tree_head_v2: SignedTreeHeadDataV2;
        case consistency_proof_v2: ConsistencyProofDataV2;
        case inclusion_proof_v2: InclusionProofDataV2;
        case x509_sct_with_proof_v2: SCTWithProofDataV2;
        case precert_sct_with_proof_v2: SCTWithProofDataV2;
        case BIN_entry_v2: TimestampedBinaryEntryDataV2;
        case BIN_sbt_v2: SignedBinaryTimestampDataV2;
        case BIN_sbt_with_proof_v2: SBTWithProofDataV2;
    } data;
} TransItem;
```
New Merkle Tree Leaves

opaque TBSSignedSoftware<1..2^24-1>;
struct {
    uint64 timestamp;
    opaque issuer_key_hash<32..2^8-1>;
    BIN_Signed_Type bin_signed_type;
    TBSSignedSoftware tbs_signed_software;
    // the DER encoded TBSSignedSoftware from the "signed_software"
    SbtExtension sbt_extensions<0..2^16-1>;
} TimestampedBinaryEntryDataV2;
New Structure of the Signed Binary Timestamp

• An SBT is a “TransItem” structure of type “bin_sbt_v2”, which encapsulates a "SignedBinary TimestampDataV2" structure:

```c
enum {
    reserved(65535)
} SbtExtensionType;

struct {
    SbtExtensionType sbt_extension_type;
    opaque sbt_extension_data<0..2^16-1>;
} SbtExtension;

struct {
    LogID log_id;
    uint64 timestamp;
    SbtExtension sbt_extensions<0..2^16-1>;
    digitally-signed struct {
        TransItem timestamped_entry;
    } signature; // The encoding of the digitally-signed element is defined in [RFC5246].
} SignedBinaryTimestampDataV2;
```
Modified Log Client Messages

- **A new message: Add Binary Code and Certificate Chain to Log**
  POST https://<log server>/ct/v1/add-Binary-chain
  **Inputs:**
  - `bin_signed_type`: binary code or its digest
  - `software`: the binary code (or digest), the signature, and the information encapsulated in CMS[RFC5652];
  - `chain`: An array of base64-encoded certificates.
  **Outputs:**
  - `sbt`: Signed Binary Timestamp. A base64 encoded “TransItem” of type “BIN_sbt_v2”, signed by this log, that corresponds to the submitted software.

- **An extended message: Retrieve Entries and STH from Log**
  GET https://<log server>/ct/v2/get-entries
  **Inputs:**
  - `start`: 0-based index of first entry to retrieve, in decimal.
  - `end`: 0-based index of last entry to retrieve, in decimal.
  **Outputs:**
  - `entries`: An array of objects, each consisting of
    - `leaf_input`: The base64 encoded "TransItem" structure of type ... or "BIN_entry_v2".
    - `log_entry`: The base64 encoded log entry. ... in the case of a "BIN_entry_v2", this is the whole "BinaryChainEntryV2".
    - `sbt`: The base64 encoded "TransItem" of ... or "BIN_sbt_v2" corresponding to this log entry.
    - `sth`: A base64 encoded “TransItem” of type “signed_tree_head_v2”, signed by this log.

Error codes:
Be identical with the according part in Section 5. 1 (Add Chain to Log) of [I-D.ietf-trans-rfc6962-bis].
Discussion

• Comments are welcome!

• Keep on improving...
Thanks!

Liang Xia (Frank)