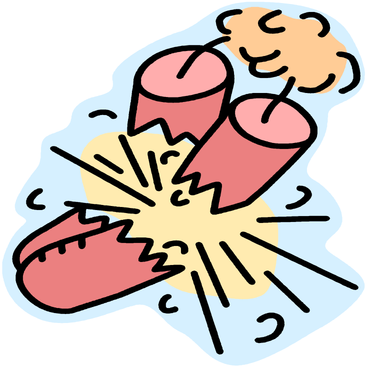
IEEE STANDARDS ASSOCIATION

IEEE Software Taggant System in Action

Igor Muttik, McAfee Labs

Mark Kennedy, Symantec



> “A **taggant** is a chemical or physical marker added to materials to allow various forms of testing. Taggants allow testing marked items for qualities such as lot number and concentration (to test for dilution, for example). In particular, taggants are known to be widely used in plastic, sheet and flexible explosives.”

http://en.wikipedia.org/wiki/Taggant

Problem of packed malware

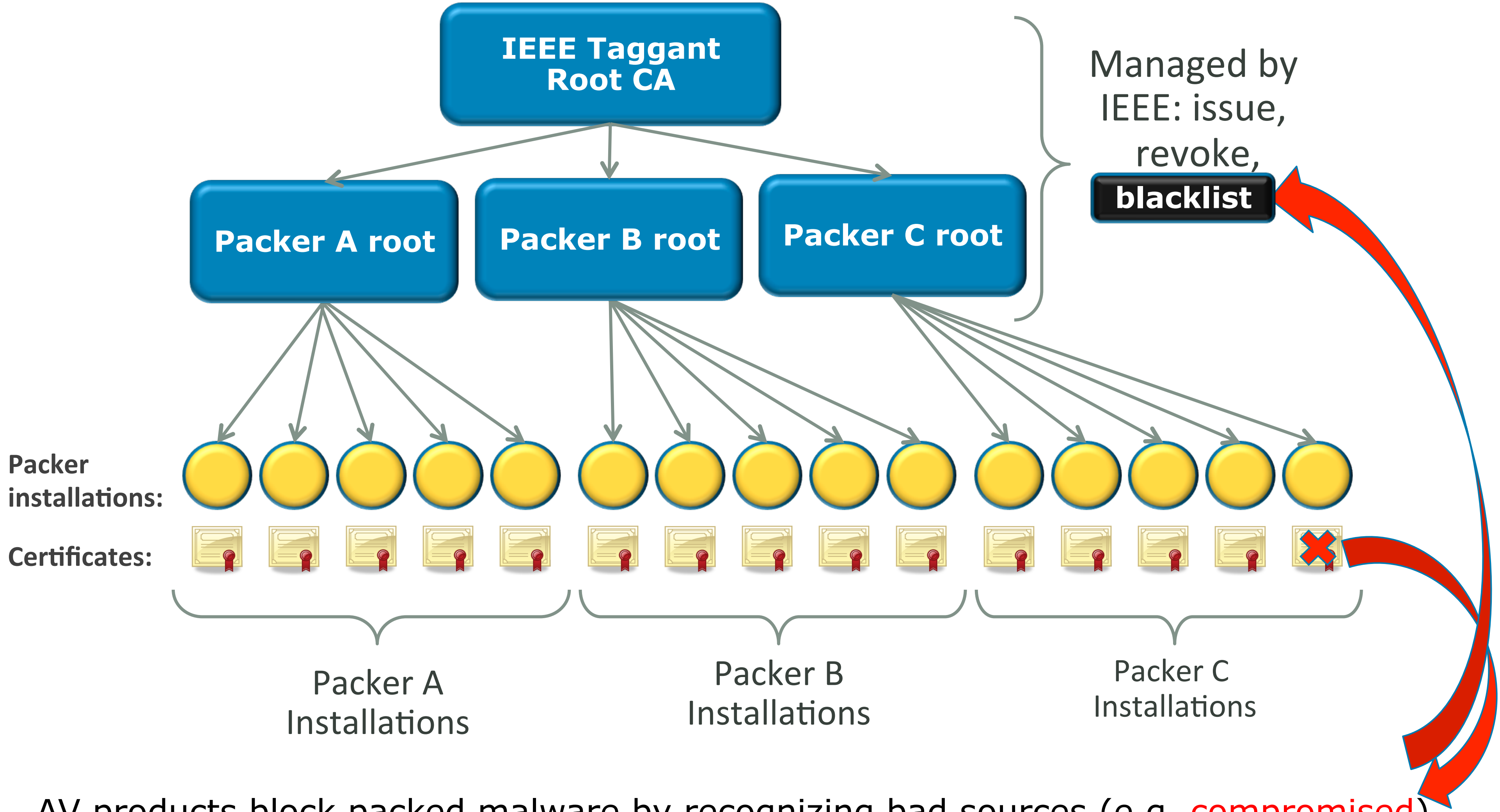
- At least 50% of malware is packed and a big headache for AV companies
- A major source of server-side polymorphics common in the Internet

- Would it not be nice to remove this source of malware?

Benefits of the system

- Security Vendors
  - More proactive protection
  - Less false positives and slowdowns
  - Less resources wasted
- Software Packer Vendors
  - Less false positives
  - Enforcing of licensing, less piracy, higher returns
  - One point of contact with security industry
  - SPV are now part of the solution
  - Competitive benefits
  - It is free
- Packer Users and End-Users
  - Less false positives and slowdowns
  - It is transparent and free (unlike digital signatures)
- We are hoping to solve the problem of packed malware in ~2-3 years

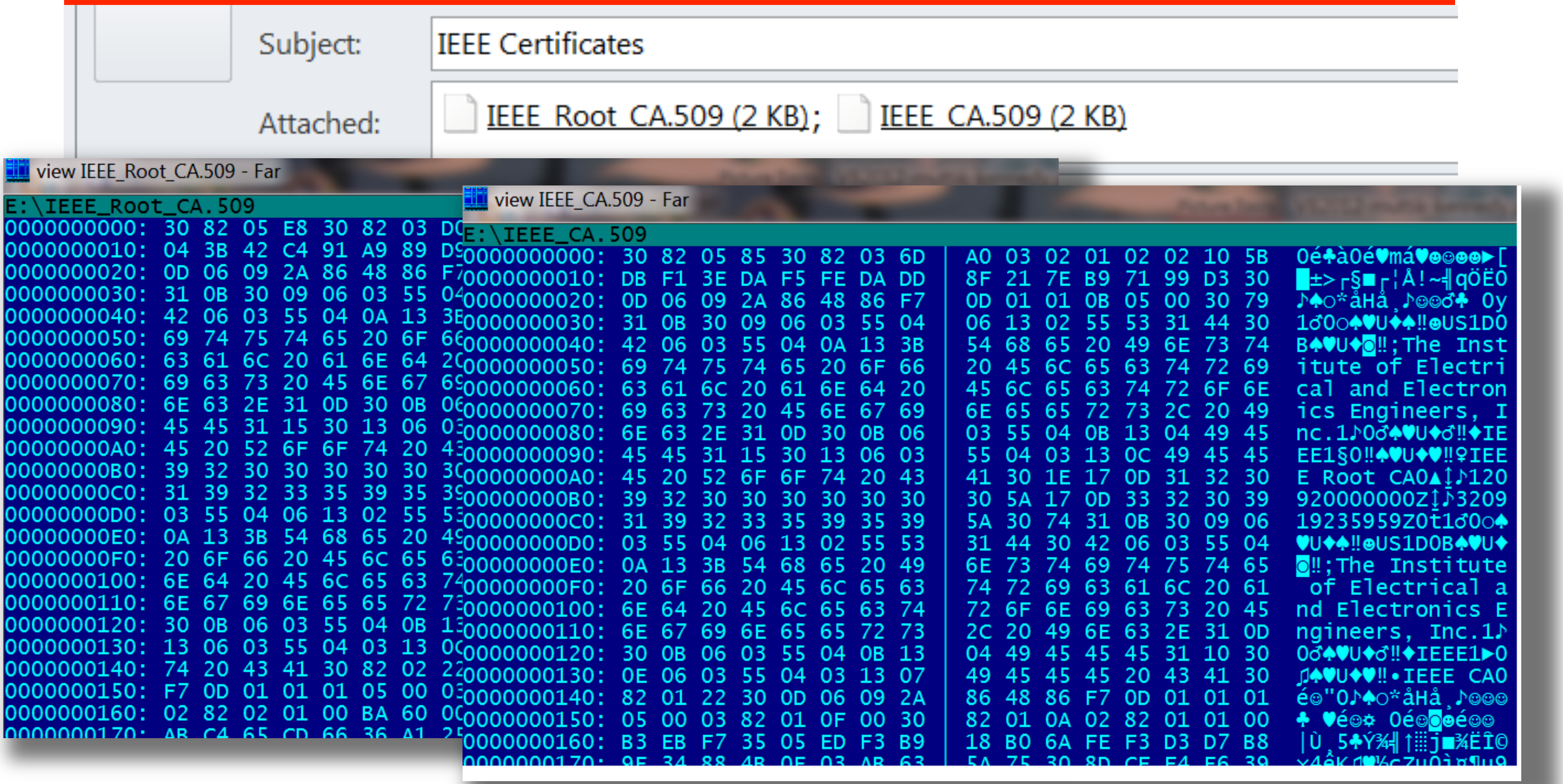
How the System Works



AV products block packed malware by recognizing bad sources (e.g. compromised)

IEEE root X.509 certificates

Generated at a key ceremony on 20 Sep 2012



Subject: IEEE Certificates

Attached: IEEE_Root_CA.509 (2 KB); IEEE_CA.509 (2 KB)

view IEEE_Root_CA.509 - Far

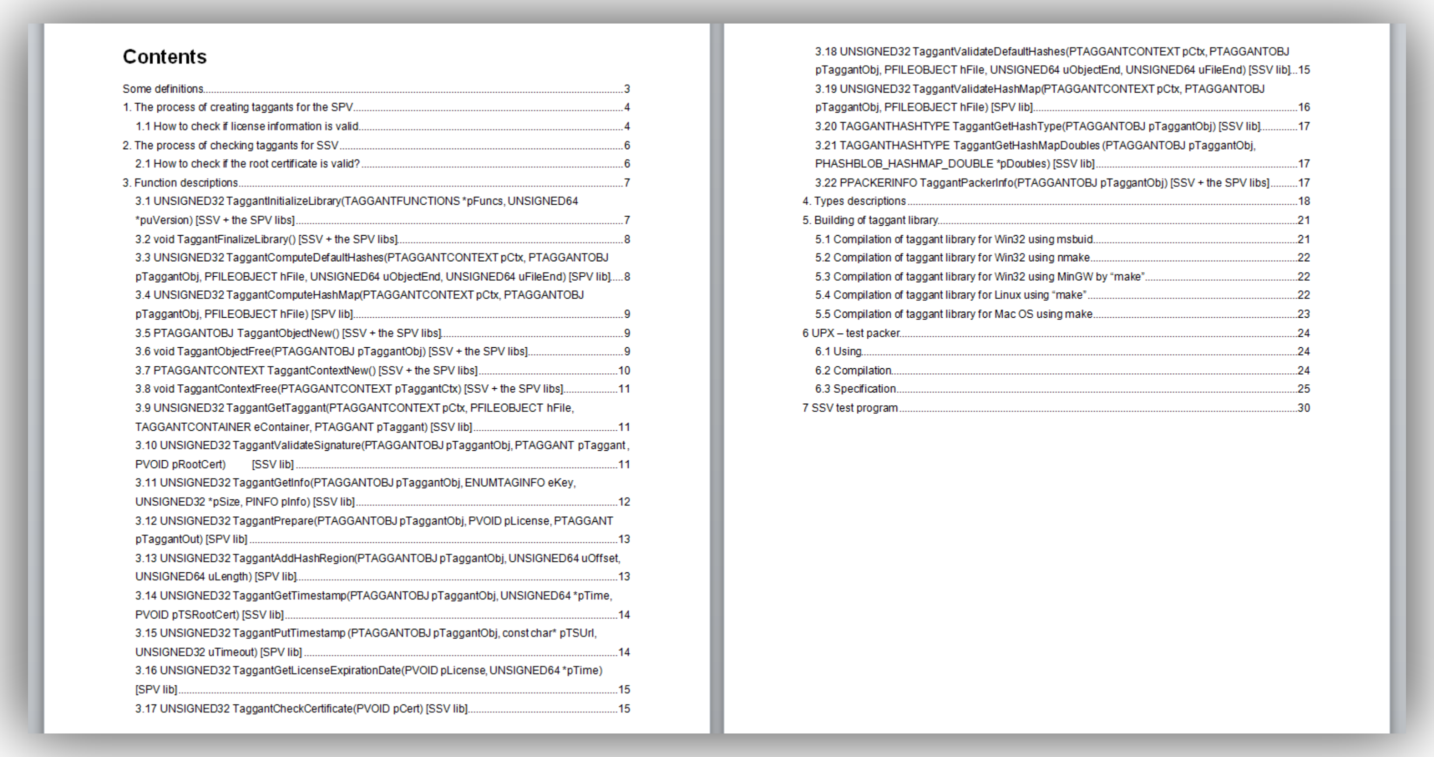
```
E:\IEEE_Root_CA.509
0000000000:  30 82 05 E8 30 82 03 D0
0000000010:  04 3B 42 C4 91 A9 89 D9
0000000020:  0D 06 09 2A 86 48 86 F7
0000000030:  31 0B 30 09 06 03 55 04
0000000040:  42 06 03 55 04 0A 13 3B
0000000050:  69 74 75 74 65 20 6F 66
0000000060:  63 61 6C 20 61 6E 64 20
0000000070:  69 63 73 20 45 6E 67 69
0000000080:  6E 63 2E 31 0D 30 0B 06
0000000090:  45 45 31 15 30 13 06 03
00000000A0:  45 20 52 6F 6F 74 20 43
00000000B0:  39 32 30 30 30 30 30 30
00000000C0:  31 39 32 33 35 39 35 39
00000000D0:  03 55 04 06 13 02 55 53
00000000E0:  0A 13 3B 54 68 65 20 49
00000000F0:  20 6F 66 20 45 6C 65 63
0000000100:  6E 64 20 45 6C 65 63 74
0000000110:  6E 67 69 6E 65 65 72 73
0000000120:  30 0B 06 03 55 04 0B 13
0000000130:  13 06 03 55 04 03 13 0C
0000000140:  74 20 43 41 30 82 02 22
0000000150:  F7 0D 01 01 01 05 00 03
0000000160:  02 82 02 01 00 BA 60 00
0000000170:  AB C4 65 CD 66 36 A1 25
```

view IEEE_CA.509 - Far

```
E:\IEEE_CA.509
0000000000:  30 82 05 85 30 82 03 6D │ A0 03 02 01 02 02 10 5B  0é♣à0é♥má♥☻☺☻☻►[
0000000010:  DB F1 3E DA F5 FE DA DD │ 8F 21 7E B9 71 99 D3 30  █±>┌§■┌¦Å!~╣qÖË0
0000000020:  0D 06 09 2A 86 48 86 F7 │ 0D 01 01 0B 05 00 30 79  ♪♠○*åHå¸♪☺☺♂♣ 0y
0000000030:  31 0B 30 09 06 03 55 04 │ 06 13 02 55 53 31 44 30  1♂0○♠♥U♦♠‼☻US1D0
0000000040:  42 06 03 55 04 0A 13 3B │ 54 68 65 20 49 6E 73 74  B♠♥U♦◙‼;The Inst
0000000050:  69 74 75 74 65 20 6F 66 │ 20 45 6C 65 63 74 72 69  itute of Electri
0000000060:  63 61 6C 20 61 6E 64 20 │ 45 6C 65 63 74 72 6F 6E  cal and Electron
0000000070:  69 63 73 20 45 6E 67 69 │ 6E 65 65 72 73 2C 20 49  ics Engineers, I
0000000080:  6E 63 2E 31 0D 30 0B 06 │ 03 55 04 0B 13 04 49 45  nc.1♪0♂♠♥U♦♂‼♦IE
0000000090:  45 45 31 15 30 13 06 03 │ 55 04 03 13 0C 49 45 45  EE1§0‼♠♥U♦♥‼♀IEE
00000000A0:  45 20 52 6F 6F 74 20 43 │ 41 30 1E 17 0D 31 32 30  E Root CA0▲↨♪120
00000000B0:  39 32 30 30 30 30 30 30 │ 30 5A 17 0D 33 32 30 39  920000000Z↨♪3209
00000000C0:  31 39 32 33 35 39 35 39 │ 5A 30 74 31 0B 30 09 06  19235959Z0t1♂0○♠
00000000D0:  03 55 04 06 13 02 55 53 │ 31 44 30 42 06 03 55 04  ♥U♦♠‼☻US1D0B♠♥U♦
00000000E0:  0A 13 3B 54 68 65 20 49 │ 6E 73 74 69 74 75 74 65  ◙‼;The Institute
00000000F0:  20 6F 66 20 45 6C 65 63 │ 74 72 69 63 61 6C 20 61   of Electrical a
0000000100:  6E 64 20 45 6C 65 63 74 │ 72 6F 6E 69 63 73 20 45  nd Electronics E
0000000110:  6E 67 69 6E 65 65 72 73 │ 2C 20 49 6E 63 2E 31 0D  ngineers, Inc.1♪
0000000120:  30 0B 06 03 55 04 0B 13 │ 04 49 45 45 45 31 10 30  0♂♠♥U♦♂‼♦IEEE1►0
0000000130:  0E 06 03 55 04 03 13 07 │ 49 45 45 45 20 43 41 30  ♫♠♥U♦♥‼•IEEE CA0
0000000140:  82 01 22 30 0D 06 09 2A │ 86 48 86 F7 0D 01 01 01  é☺"0♪♠○*åHå¸♪☺☺☺
0000000150:  05 00 03 82 01 0F 00 30 │ 82 01 0A 02 82 01 01 00  ♣ ♥é☺☼ 0é☺◙☻é☺☺
0000000160:  B3 EB F7 35 05 ED F3 B9 │ 18 B0 6A FE F3 D3 D7 B8  │Ù¸5♣Ý¾╣↑▒j■¾ËÎ©
0000000170:  9E 34 88 4B 0E 03 AB 63 │ 5A 75 30 8D CE F4 F6 39
```

Status of the project – READY

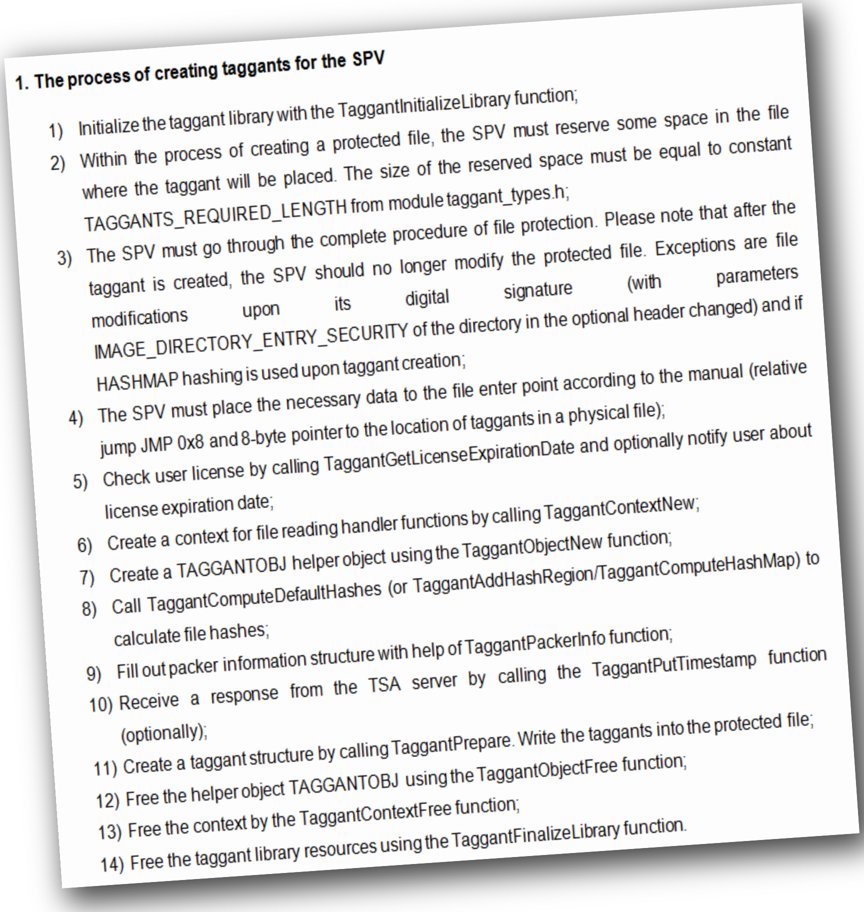
- The library based on Open SSL is ready, code reviewed and tested
  - API documentation is available
  - Includes a modified version of UPX which supports taggants



Contents

- PKI servers by VeriSign/Symantec (support blacklisting and time-stamping)

Documentation is ready



1. The process of creating taggants for the SPV

1) Initialize the taggant library with the TaggantInitializeLibrary function;

2) Within the process of creating a protected file, the SPV must reserve some space in the file where the taggant will be placed. The size of the reserved space must be equal to constant TAGGANTS_REQUIRED_LENGTH from module taggant_types.h;

3) The SPV must go through the complete procedure of file protection. Please note that after the taggant is created, the SPV should no longer modify the protected file. Exceptions are file modifications upon its digital signature (with parameters IMAGE_DIRECTORY_ENTRY_SECURITY of the directory in the optional header changed) and if HASHMAP hashing is used upon taggant creation;

4) The SPV must place the necessary data to the file enter point according to the manual (relative jump JMP 0x8 and 8-byte pointer to the location of taggants in a physical file);

5) Check user license by calling TaggantGetLicenseExpirationDate and optionally notify user about license expiration date;

6) Create a context for file reading handler functions by calling TaggantContextNew;

7) Create a TAGGANTOBJ helper object using the TaggantObjectNew function;

8) Call TaggantComputeDefaultHashes (or TaggantAddHashRegion/TaggantComputeHashMap) to calculate file hashes;

9) Fill out packer information structure with help of TaggantPackerInfo function;

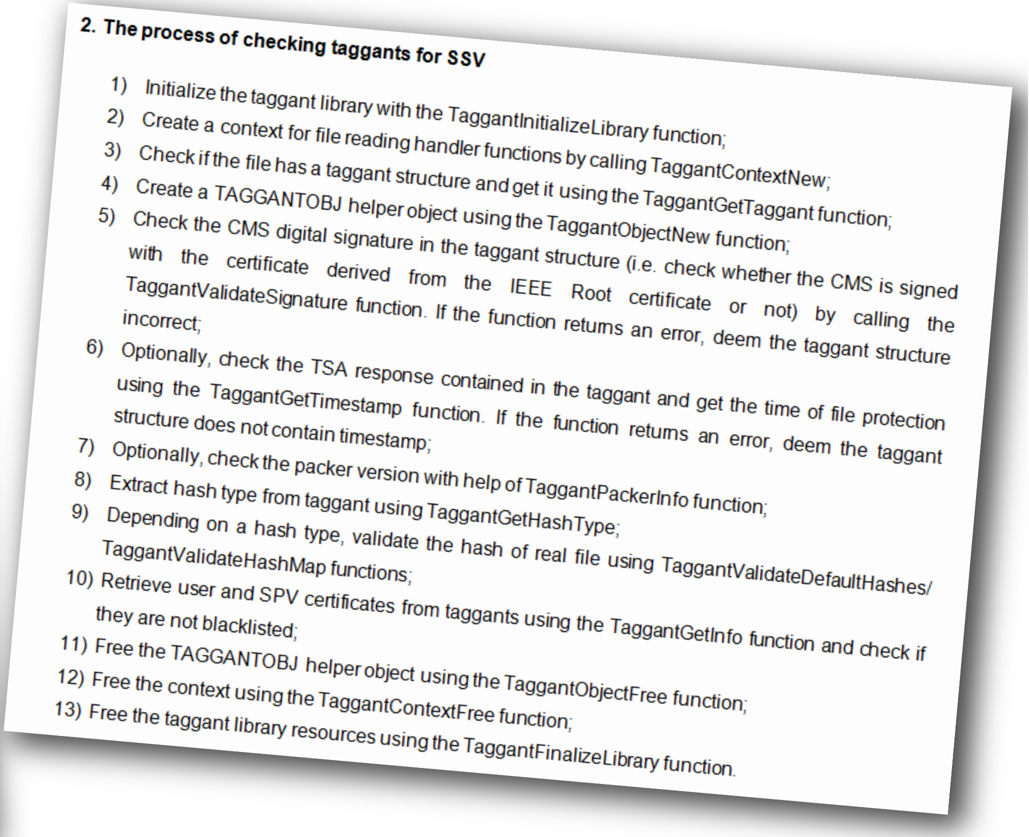
10) Receive a response from the TSA server by calling the TaggantPutTimestamp function (optionally);

11) Create a taggant structure by calling TaggantPrepare. Write the taggants into the protected file;

12) Free the helper object TAGGANTOBJ using the TaggantObjectFree function;

13) Free the context by the TaggantContextFree function;

14) Free the taggant library resources using the TaggantFinalizeLibrary function.



2. The process of checking taggants for SSV

1) Initialize the taggant library with the TaggantInitializeLibrary function;

2) Create a context for file reading handler functions by calling TaggantContextNew;

3) Check if the file has a taggant structure and get it using the TaggantGetTaggant function;

4) Create a TAGGANTOBJ helper object using the TaggantObjectNew function;

5) Check the CMS digital signature in the taggant structure (i.e. check whether the CMS is signed with the certificate derived from the IEEE Root certificate or not) by calling the TaggantValidateSignature function. If the function returns an error, deem the taggant structure incorrect;

6) Optionally, check the TSA response contained in the taggant and get the time of file protection using the TaggantGetTimestamp function. If the function returns an error, deem the taggant structure does not contain timestamp;

7) Optionally, check the packer version with help of TaggantPackerInfo function;

8) Extract hash type from taggant using TaggantGetHashType;

9) Depending on a hash type, validate the hash of real file using TaggantValidateDefaultHashes/ TaggantValidateHashMap functions;

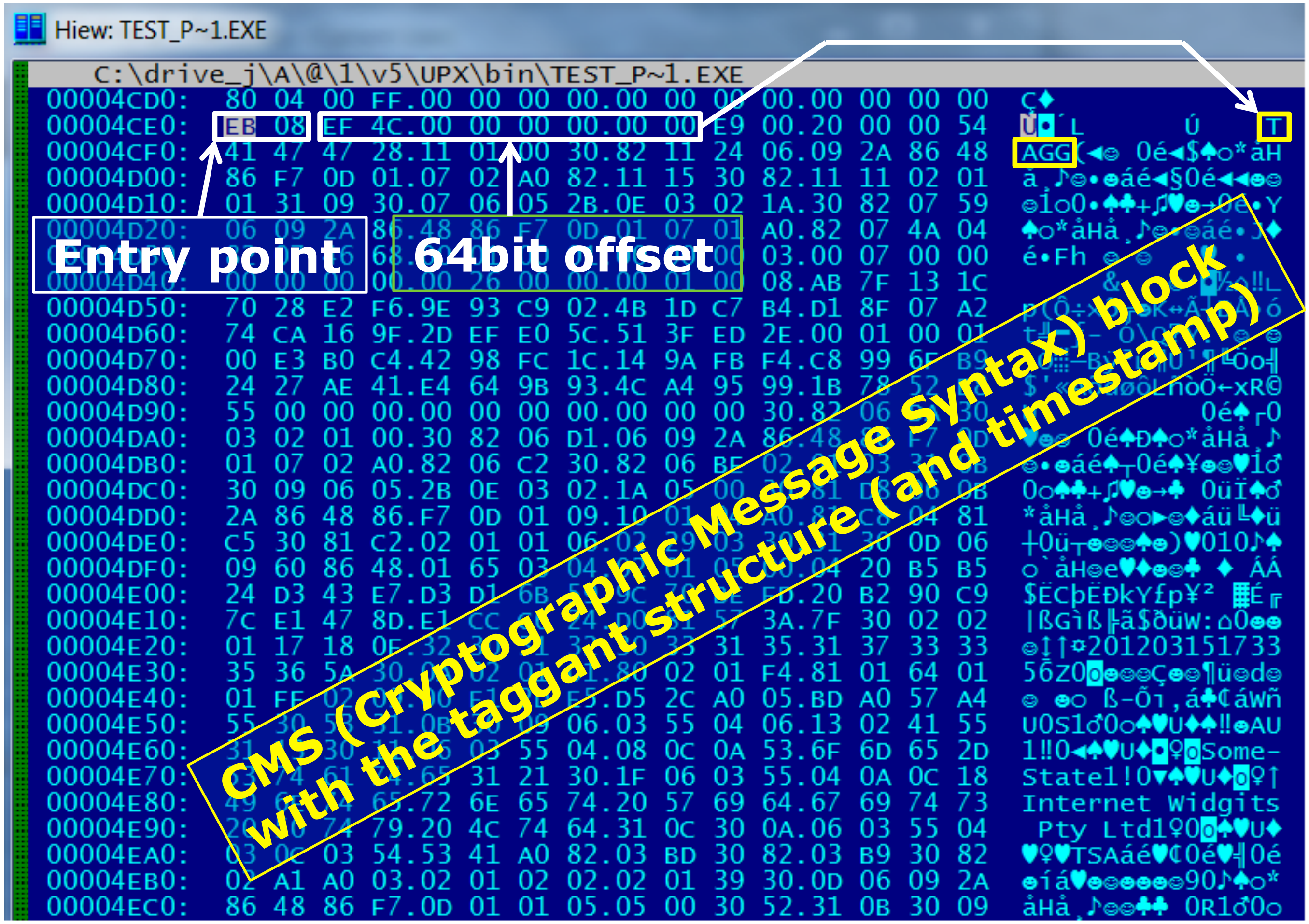
10) Retrieve user and SPV certificates from taggants using the TaggantGetInfo function and check if they are not blacklisted;

11) Free the TAGGANTOBJ helper object using the TaggantObjectFree function;

12) Free the context using the TaggantContextFree function;

13) Free the taggant library resources using the TaggantFinalizeLibrary function.

Taggant_enabled_UPX(CALC.EXE)



Hiew: TEST_P~1.EXE

```
   C:\drive_j\A\@\1\v5\UPX\bin\TEST_P~1.EXE
00004CD0:  80 04 00 FF.00 00 00 00.00 00 00 00.00 00 00 00  Ç♦
00004CE0:  EB 08 EF 4C.00 00 00 00.00 00 E9 00.20 00 00 54  Ù◘´L      Ú   T
00004CF0:  41 47 47 28.11 01 00 30.82 11 24 06.09 2A 86 48  AGG(◄☺ 0é◄$♠○*åH
00004D00:  86 F7 0D 01.07 02 A0 82.11 15 30 82.11 11 02 01  å≈♪☺•☻áé◄§0é◄◄☻☺
00004D10:  01 31 09 30.07 06 05 2B.0E 03 02 1A.30 82 07 59  ☺1○0•♠♣+♫♥☻→0é•Y
00004D20:  06 09 2A 86.48 86 F7 0D.01 07 01 A0.82 07 4A 04  ♠○*åHå≈♪☺•☺áé•J♦
00004D30:  [illegible] 68.[illegible] 00 03.00 07 00 00  é•Fh ☺☺  •
00004D40:  00 00 00 00.00 26 00 00.00 01 00 08.AB 7F 13 1C     &   ☺ ◘½⌂‼∟
00004D50:  70 28 E2 F6.9E 93 C9 02.4B 1D C7 B4.D1 8F 07 A2  p(Ô÷×ô╔☻K↔Ã┤ÐÅ•ó
00004D60:  74 CA 16 9F.2D EF E0 5C.51 3F ED 2E.00 01 00 01  t╩▬ƒ-´Ó\Q?Ý. ☺ ☺
00004D70:  00 E3 B0 C4.42 98 FC 1C.14 9A FB F4.C8 99 6F B9   Ò░─Bÿ³∟¶Ü¹¶╚Öo╣
00004D80:  24 27 AE 41.E4 64 9B 93.4C A4 95 99.1B 78 52 [illegible]  $'«AõdøôLñòÖ←xR©
00004D90:  55 00 00 00.00 00 00 00.00 00 00 30.82 06 [illegible] 30  U           0é♠┌0
00004DA0:  03 02 01 00.30 82 06 D1.06 09 2A 86.48 [illegible] F7 0D  ♥☻☺ 0é♠Ð♠○*åHå≈♪
00004DB0:  01 07 02 A0.82 06 C2 30.82 06 BE 02.[illegible] 03 31 0B  ☺•☻áé♠┬0é♠¥☻☺♥1♂
00004DC0:  30 09 06 05.2B 0E 03 02.1A 05 00 [illegible].81 D8 06 0B  0○♠♣+♫♥☻→♣ 0üÏ♠♂
00004DD0:  2A 86 48 86.F7 0D 01 09.10 01 [illegible] A0.81 C8 04 81  *åHå≈♪☺○►☺♦áü╚♦ü
00004DE0:  C5 30 81 C2.02 01 01 06.02 29 03 30.31 30 0D 06  ┼0ü┬☻☺☺♠☻)♥010♪♠
00004DF0:  09 60 86 48.01 65 03 04.[illegible] 01 05 00.04 20 B5 B5  ○`åH☺e♥♦☻☺♣ ♦ ÁÁ
00004E00:  24 D3 43 E7.D3 D1 6B 59.9C 70 BE FD.20 B2 90 C9  $ËCþËÐkY£p¥² ▓É╔
00004E10:  7C E1 47 8D.E1 CC [illegible] 24.D0 81 57 3A.7F 30 02 02  |ßGìß╠ã$ðüW:⌂0☻☻
00004E20:  01 17 18 0F.32 [illegible] 31 32.30 33 31 35.31 37 33 33  ☺↨↑☼201203151733
00004E30:  35 36 5A 30.[illegible] 02 01 01.80 02 01 F4.81 01 64 01  56Z0◙☺☺☺Ç☺☺¶ü☺d☺
00004E40:  01 FF 02 09.00 E1 [illegible] E5.D5 2C A0 05.BD A0 57 A4  ☺ ☻○ ß-Õı,á♣¢áWñ
00004E50:  55 30 53 31.0B 30 09 06.03 55 04 06.13 02 41 55  U0S1♂0○♠♥U♦♠‼☻AU
00004E60:  31 13 30 11.06 03 55 04.08 0C 0A 53.6F 6D 65 2D  1‼0◄♠♥U♦◘♀◙Some-
00004E70:  53 74 61 74.65 31 21 30.1F 06 03 55.04 0A 0C 18  State1!0▼♠♥U♦◙♀↑
00004E80:  49 6E 74 65.72 6E 65 74.20 57 69 64.67 69 74 73  Internet Widgits
00004E90:  20 50 74 79.20 4C 74 64.31 0C 30 0A.06 03 55 04   Pty Ltd1♀0◙♠♥U♦
00004EA0:  03 0C 03 54.53 41 A0 82.03 BD 30 82.03 B9 30 82  ♥♀♥TSAáé♥¢0é♥╣0é
00004EB0:  02 A1 A0 03.02 01 02 02.02 01 39 30.0D 06 09 2A  ☻íá♥☻☺☻☻☻☺90♪♠○*
00004EC0:  86 48 86 F7.0D 01 01 05.05 00 30 52.31 0B 30 09  åHå≈♪☺☺♣♣ 0R1♂0○
```

Entry point

64bit offset

CMS (Cryptographic Message Syntax) block with the taggant structure (and timestamp)

You will soon see packed files with taggants (**EB 08** at the entry point +”**TAGG**”)

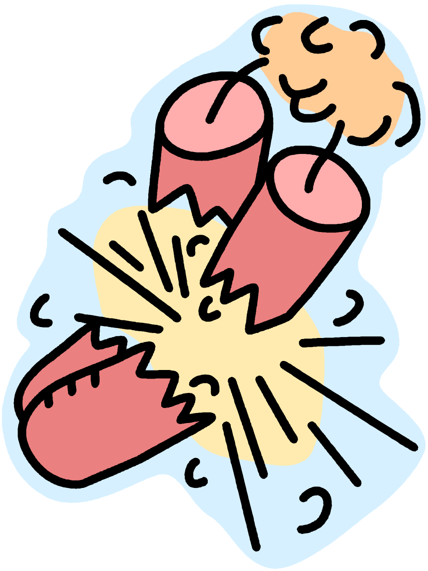
Using the system

- If you:
  - Want to check the taggant validity
  - Want to use the taggant library to parse the taggant CMS structure
  - Want to check that the certificate is not blacklisted (packer installation is a valid packer customer)
  - Want to participate in blacklisting of packed malware sources

Summary

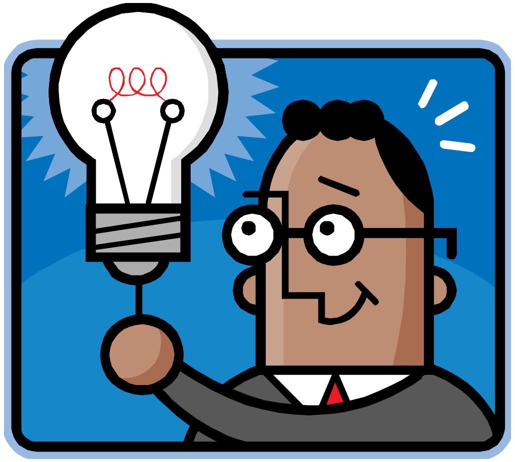
- The system is ready to go
- You will see packed files with taggants soon
- To be able to crack open, verify the CMS structure and check the black list you will need to licence the system

The End



1. The proceedings contain full API guide
2. http://standards.ieee.org/develop/indconn/icsg
3. https://media.blackhat.com/bh-us-11/Kennedy/ BH_US_11_KennedyMuttik_IEEE_Slides.pdf

Questions, please



Backup slides

Taggant vs authenticode

- Taggant contains a “performant” hash (SHA256 by default)
  - Covers only vital executable areas
- Taggant allows a fall-back on to a “default” hash
  - It covers the whole file (almost whole)
  - Will be used if the performant hash is broken
- Creating and using files with taggants is **free**
  - Included by the packing software automatically
  - The PKI infrastructure will be sponsored by AV companies
- Taggants are compatible with authenticode
  - Digital signature can be applied after a packer included a taggant

The lifecycle

Step 1 – packer vendor

- New packer vendor contacts IEEE
- IEEE verifies the vendor
- IEEE creates a vendor login
- Vendor asks for a URL for a user
- URL is embedded into the license for each user’s packer setup
- Packer user gets the packer setup

Step 2 – packer software setup

- The setup logs into a unique URL
- IEEE creates a key pair
- Setup gets a certificate back

Step 3 – packer obfuscates a file

- Packer is executed to pack a file
- Taggant is created with 3 hashes
- Timestamp is included
- Setup/user certificate is included
- Taggant is part of the packed file
- Packed file is distributed

Step 4 – packed file executes

- End user runs a packed file
- AV checks the source (the setup certificate & maybe a timestamp)
- AV blocks if bad