

俺とお前と Certificate Transparency



すみだセキュリティ勉強会 2015#1
2015/05/09 by @ozuma5119

「俺とお前とCertificate Transparency」というタイトルで発表します、すみだセキュリティ勉強会主催のozuma5119です。

@ozuma5119

- セキュリティっぽいITエンジニア(pentester)
- セキュリティっぽいBlog :ろば電子が詰まっている
 - <http://d.hatena.ne.jp/ozuma/>
- 科学写真家(と名乗っている)

2

すみだセキュリティ勉強会を主催しています、ozuma5119と申します。
ふだんは、比較的固めの会社でセキュリティエンジニアをしています。ブログはこちら。

<http://d.hatena.ne.jp/ozuma/>

科学写真家というのは、「理科の教科書に載ってるような写真」を撮る人たちです。
こちらは副業ということで、小学生向けの教材の写真とか撮って、ときどき本に載ったりします。

きょう、話すことの概要

[前座] 証明書の鑑賞方法（素数）

[メイン] Certificate Transparency

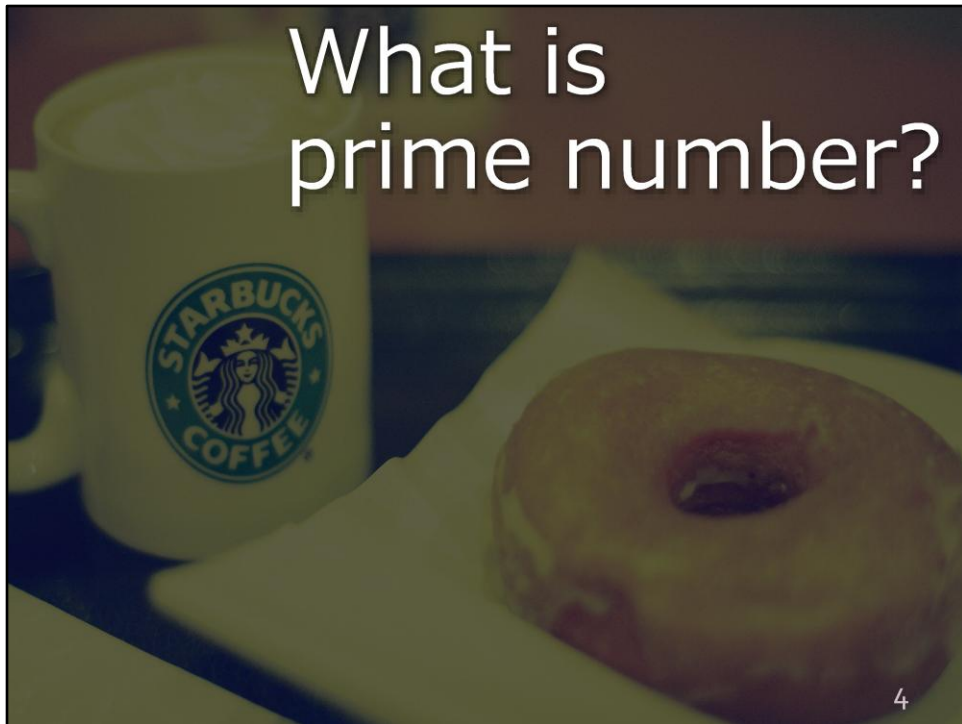
3

今回の概要はこんな感じです。

今日のメインは、2つめのCertificate Transparencyというものです。これについて話そうと思うのですが、その前にまずSSL証明書の鑑賞方法を学んでおきましょう。CTを理解するには、証明書の見方を知らないといけないためです。

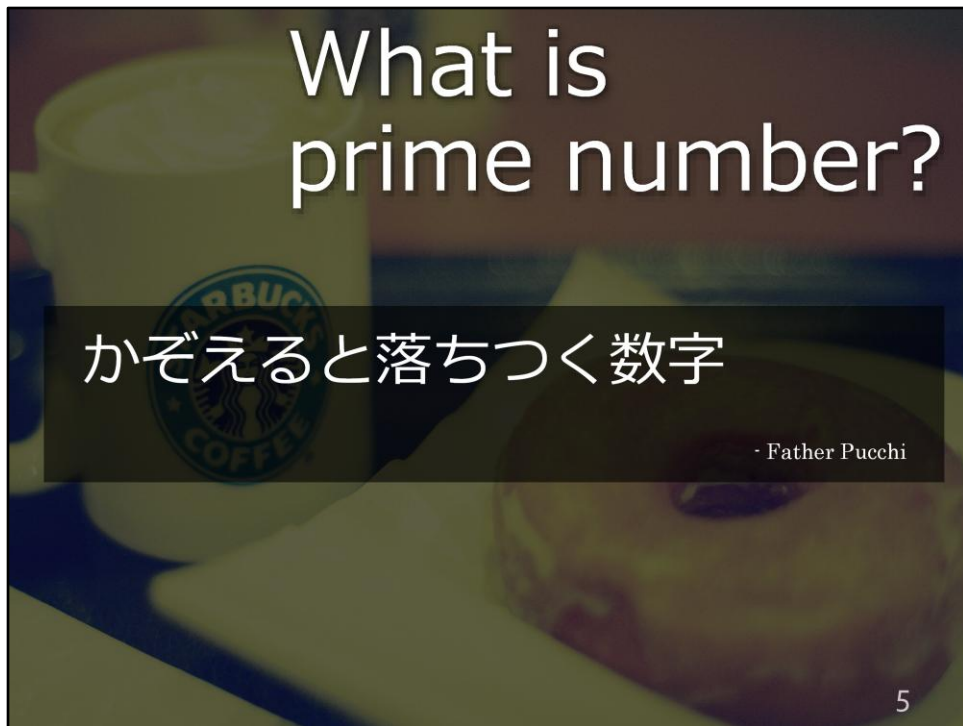
ただ、単に証明書を見ても、証明書マニアじゃないとつらいと思うので、今日は素数に注目して証明書を見てみましょう。

素数に注目して証明書を見る→見方が分かったところでCTを理解する、という流れです。



で、まず素数についてです。英語ではprime numberと言います。RSA公開鍵の中身を見たりするとprimeという言葉が出てくるので、慣れておきましょう

素数。素数ってなんでしょう？ まあこの勉強会に来るような方なら皆さんご存じと思いますが.....



はい。素数とは、かぞえると落ち着く数ですね。
Father Pucchiが言っていましたから間違いありません。

復習：「公開鍵ならば素数」じゃないよ

公開鍵
暗号方式

- RSA (2つの素数の積)
- 楕円曲線 (離散対数問題)
-

でも今日は素数に注目したいので
RSAの証明書

6

証明書の公開鍵を見る前に、一点、確認しておきましょう。

現在多くのSSL証明書では、公開鍵暗号方式としてRSAを利用しています。これは比較的有名な、素因数分解の困難さを利用したものです。

でも、公開鍵暗号方式にはいくつかの実装があり、実際最近では、楕円曲線によるものも増えつつあります。

が、今日は現在もっともポピュラーな、素数を利用したRSAの証明書を見ます。

復習：証明書の確認方法



証明書の見方を復習しておきましょう。ブラウザによって若干異なりますが、多くの場合には鍵マークをクリックして証明書を表示できます。

RSAを利用した公開鍵証明書ならば、このように公開キーとして現在は一般的に2048bitの数値が入っています。

次のOpenSSLコマンドを利用した例で、もうちょっと証明書について細かく見てみましょう。

```
echo Q | openssl s_client -connect dena.com:443 | openssl x509 -text
```

Certificate:

Data:

Version: 3 (0x2)

Serial Number:

03:07:f6:39:b3:34:b3:c2:94:32:05:fc:4f:2e:6b:of

Signature Algorithm: sha256WithRSAEncryption

Issuer: C=US, O=Symantec Corporation,

OU=Symantec Trust Network, CN=Symantec Class 3
Secure Server CA - G4

Validity

Not Before: Feb 17 00:00:00 2015 GMT

Not After : Mar 30 23:59:59 2016 GMT

Subject: C=JP, ST=Tokyo, L=Shibuya-ku, O=DeNA
Co., Ltd., OU=Head office 1, CN=dena.com

Subject Public Key Info:

Public Key Algorithm: rsaEncryption

RSA Public Key: (2048 bit)

Modulus (2048 bit):

00:af:1d:59:df:c8:4f:50:ae:48:2b:b8:25:d4:51:
a4:ea:81:5b:do:ec:4d:64:d8:a1:da:db:c6:fd:af:
3e:ba:0a:8c:02:70:0c:38:5c:74:06:ce:bb:38:cb:
2b:ac:df:a1:23:51:f7:a6:c5:b3:2a:f6:f4:7e:62:
34:a7:de:cb:ea:2a:c4:b8:00:c2:f8:01:15:58:45:

3f:04:76:21:2d:1c:dd:18:cb:38:fe:ea:bo:82:b9:
87:3a:3b:d8:48:9e:c6:a2:92:9b:09:78:df:93:09:
65:ee:21:e1:dd:b4:4d:fd:52:b8:5b:01:e7:1c:da:
8b:ff:bc:c9:40:cc:17:73:64:e6:ab:78:b5:84:bc:
97:82:95:8f:71:84:1a:e7:77:f9:0c:03:1b:fc:bd:
b9:27:91:46:1f:5e:51:aa:64:46:27:ao:2a:36:38:
of:52:83:ed:73:ae:f6:f3:3a:d3:9a:a8:a7:ca:c9:
46:b5:0b:ca:8e:34:76:e3:c5:5b:dd:34:b9:b3:ca:
69:90:53:ea:e7:94:7f:8c:82:00:61:80:aa:1a:da:
7e:f3:7e:84:fa:e6:b1:3f:07:3f:0a:2e:5b:a3:35:
e1:35:af:16:b7:41:85:eb:0e:31:ea:40:73:a3:e7:
20:40:d5:ea:c3:eb:1a:cc:d2:7d:92:15:45:7a:bo:
3d:25

Exponent: 65537 (0x10001)

X509v3 extensions:

X509v3 Subject Alternative Name:

DNS:dena.com

X509v3 Basic Constraints:

CA:FALSE

.....

8

証明書はOpenSSLコマンドで直接見てみることもできます。慣れているひとなら、こちらの方がいいでしょう。

```
echo Q | openssl s_client -connect dena.com:443 | openssl x509 -text
```

はじめにecho Qしているのは、普通につなぐとコネクションが張りっぱなしになるのでいきなりQUITするためです。まあ手でやるならCtrl+C押せばいいので不要ですが、シェルスクリプトなどで証明書を集めたいときはこういうことをすると良いでしょう。


```
echo Q | openssl s_client -connect dena.com:443 | openssl x509 -text
```

Certificate:

Data:

Version: 3 (0x2)

Serial Number:

03:07:f6:39:b3:34:b3:c2:94:32:05:fc:4f:2e:6b

Signature Algorithm: sha256WithRSAEncryption

Issuer: C=US, O=Symantec Corporation,
OU=Symantec Trust Network, CN=Symantec Class 3 Secure Server CA - G4

Validity

Not Before: Feb 17 00:00:00 2015 GMT

Not After : Mar 30 23:59:59 2016 GMT

Subject: C=JP, ST=Tokyo, L=Shibuya-ku, O=DeNA Co., Ltd., OU=Head office 1, CN=dena.com

Subject Public Key Info:

Public Key Algorithm: rsaEncryption

RSA Public Key: (2048 bit)

Modulus (2048 bit):

00:af:1d:59:df:c8:4f:50:ae:48:2b:b8:25:d

a4:ea:81:5b:do:ec:4d:64:d8:a1:da:db:c6:3

3e:ba:0a:8c:02:70:0c:38:5c:74:06:ce:bb:3

2b:ac:df:a1:23:51:f7:a6:c5:b3:2a:f6:f4:7e:3

34:a7:de:cb:ea:2a:c4:b8:00:c2:f8:01:15:5

Issuer: C=US, O=Symantec Corporation,
OU=Symantec Trust Network,
CN=Symantec Class 3 Secure Server CA
- G4

Validity

Not Before: Feb 17 00:00:00 2015

GMT

Not After : Mar 30 23:59:59 2016

GMT

Subject: C=JP, ST=Tokyo,
L=Shibuya-ku, O=DeNA Co., Ltd.,
OU=Head office 1, CN=dena.com

Issuer、証明書を発行した人。

Subjectは証明書の主体者、組織名とかドメイン名が入っています。これはDeNAのサーバなので、CNにdena.comと入ってますね。

```
echo Q | openssl s_client -connect dena.com:443 | openssl x509 -text
```

Certificate:

Data:

Version: 3 (0x2)

Serial Number:

03:07:f6:39:b3:34:b3:c2:94:32:05:fc:4f:2e:6b:0f

Signature Algorithm: sha256WithRSAEncryption

Issuer: C=US, O=Symantec Corporation,

OU=Symantec Trust Networks, CN=Symantec Class 3

Secure Server

Validity

Not Bef

Not Aft

Subject: C

Co., Ltd., OU=Head Office 1, CN=dena.com

Subject Public Key Info:

Public Key Algorithm: rsaEncryption

RSA Public Key: (2048 bit)

Modulus (2048 bit):

00:af:1d:59:df:c8:4f:50:ae:48:2b:b8:25:d4:51:

a4:ea:81:5b:do:ec:4d:64:d8:a1:da:db:c6:fd:af:

3e:ba:0a:8c:02:70:0c:38:5c:74:06:ce:bb:38:cb:

2b:ac:df:a1:23:51:f7:a6:c5:b3:2a:f6:f4:7e:62:

34:a7:de:cb:ea:2a:c4:b8:00:c2:f8:01:15:58:45:

Modulus
→ 2つの素数の積

3f:04:76:21:2d:1c:dd:18:cb:38:fe:ea:bo:82:b9:
87:3a:3b:d8:48:9e:c6:a2:g2:9b:09:78:df:93:09:
65:ee:21:e1:dd:b4:4d:fd:52:b8:5b:01:e7:1c:da:
8b:ff:bc:c9:40:cc:17:73:64:e6:ab:78:b5:84:bc:
97:82:95:8f:71:84:1a:e7:77:f9:0c:03:1b:fc:bd:
b9:27:91:46:1f:5e:51:aa:64:46:27:ao:2a:36:38:
of:52:83:ed:73:ae:f6:f3:3a:d3:9a:a8:a7:ca:c9:
46:b5:0b:ca:8e:34:76:e3:c5:5b:dd:34:b9:b3:ca:
69:90:53:ea:e7:94:7f:8c:82:00:61:80:aa:1a:da:
7e:f3:7e:84:fa:e6:b1:3f:07:3f:0a:2e:5b:a3:35:
e1:35:af:16:b7:41:85:eb:0e:31:ea:40:73:a3:e7:
20:40:d5:ea:c3:eb:1a:cc:d2:7d:g2:15:45:7a:bo:
3d:25

Exponent: 65537 (0x10001)

X509v3 extensions:

X509v3 Subject Alternative Name:

DNS:dena.com

X509v3 Basic Constraints:

CA:FALSE

.....

10

Modulusが2つの素数の積ですね。

コロン区切りの16進数で書かれています。これを素因数分解できればいいわけだ。

```
echo Q | openssl s_client -connect dena.com:443 | openssl x509 -text
```

Certificate:

Data:

Version: 3 (0x2)

Serial Number:

03:07:f6:39:b3:34:b3:c2:94:32:05:fc:4f:2e:6b:of

Signature Algorithm: sha256WithRSAEncryption

Issuer: C=US, O=Symantec Corporation,

OU=Symantec Trust Network, CN=Symantec Class 3

Secure Server CA - G4

Validity

Not Before

Not After

Subject: C=J

Co., Ltd., OU=Head Office 1, CN=dena.com

Subject Public Key Info:

Public Key Algorithm: rsaEncryption

RSA Public Key: (2048 bit)

Modulus (2048 bit):

00:af:1d:59:df:c8:4f:50:ae:48:2b:b8:25:d4:51:

a4:ea:81:5b:do:ec:4d:64:d8:a1:da:db:c6:fd:af:

3e:ba:0a:8c:02:70:0c:38:5c:74:06:ce:bb:38:cb:

2b:ac:df:a1:23:51:f7:a6:c5:b3:2a:f6:f4:7e:62:

34:a7:de:cb:ea:2a:c4:b8:00:c2:f8:01:15:58:45:

3f:04:76:21:2d:1c:dd:18:cb:38:fe:ea:bo:82:b9:

87:3a:3b:d8:48:9e:c6:a2:92:9b:09:78:df:93:09:

65:ee:21:e1:dd:b4:4d:fd:52:b8:5b:01:e7:1c:da:

8b:ff:bc:c9:40:cc:17:73:64:e6:ab:78:b5:84:bc:

97:82:95:8f:71:84:1a:e7:77:f9:0c:03:1b:fc:bd:

b9:27:91:46:1f:5e:51:aa:64:46:27:ao:2a:36:38:

of:52:83:ed:73:ae:f6:f3:3a:d3:ga:a8:a7:ca:c9:

46:b5:0b:ca:8e:34:76:e3:c5:5b:dd:34:b9:b3:ca:

69:90:53:ea:e7:94:7f:8c:82:00:61:80:aa:1a:da:

7e:f3:7e:84:fa:e6:b1:3f:07:3f:0a:2e:5b:a3:35:

e1:35:af:16:b7:41:85:eb:0e:31:ea:40:73:a3:e7:

20:40:d5:ea:c3:eb:1a:cc:d2:7d:92:15:45:7a:bo:

3d:25

Exponent: 65537 (0x10001)

X509v3 extensions:

X509v3 Subject Alternative Name:

DNS:dena.com

X509v3 Basic Constraints:

CA:FALSE

X509v3 extensions

11

また、後ろにはX509v3 extensionsとして色々な拡張領域が定義されていますね。後でここも見ます。

証明書を細かく見ていくとそれだけで終わっちゃうので、次にModulusと素数の小ネタだけ話しておきましょう。

素数をいっぱい作って

modulusを素因数分解したい!

12

素数をいっぱい作って
modulusを素因数分解したい!

OpenSSLで、「出やすい素数」は無いのか?

```
$ openssl genrsa 2048
```

で素数が2つできるので、これを172万回くらい繰り返し344万個の素数をゲット（うれしい!）

```
$ wc -l prime.txt  
3440000 prime.txt
```

13

先ほどModulusを見ましたが、あれを素因数分解できれば私の勝ちなわけです。で、まっとうに素因数分解にチャレンジするやり方よりも、こういうアプローチが考えられます。

「ふつうの人はopenssl genrsaで素数を作る」→「もしこの素数生成に偏りがあれば、出やすい素数で因数分解できるmodulusが結構あるのでは?」

これは実際、今から7,8年前のDebianで発生した事例です。

とりあえずここでは、CentOS 6.6上で、344万個の素数を作りました。HDDにこんなにたくさん素数があると思うと、わくわくしますよね。

もしこの中に、何度も出ている素数があれば私の「勝ち」なわけですが.....。

「出やすい素数」

$$\begin{array}{c}
 * \qquad \qquad * \\
 * \qquad \qquad + \qquad \text{ありません} \\
 + \quad \begin{array}{c} n \wedge \\ (\exists (* \neg \wedge) E) \\ Y \qquad Y \end{array}
 \end{array}$$

14

さて、出やすい素数は.....ありませんでした。

ぜんぶuniqueだった

```
$ wc -l prime.txt
3440000 prime.txt
$ sort prime.txt | uniq | wc -l
3440000
```

ついでに国内の有名なHTTPSサイト200個ほどのmodulusを、340万個の素数で片っ端から割ってみたけど、割りきれ（素因数が見つかった）ものもなかった

15

この通り、作ったものからuniqして数えてみましたが一つも同じ素数は生成されませんでした。

まあ340万個という数は実は.....少なすぎるのです。

素数定理: 自然数x下の素数の個数 $\pi(x)$ は?

$$\pi(x) \cong \frac{x}{\ln x}$$

今、modulusが2048bitなので、
2つの素数はだいたい 2^{1024} の
オーダー。

素数候補の個数

$$\pi(2^{1024}) \cong \frac{2^{1024}}{\ln 2^{1024}} = \frac{2^{1024}}{1024 \times \ln 2} \cong 2^{1014} \times 1.4$$

16

ある自然数xが与えられたとき、そのx以下に素数がどのくらいあるか？ という素数の濃度を測る際に使われるのが素数定理です。

おおむね、素数は自然対数(eを底としたlog、lnと書く)で割った値の数となります。

いま、modulus、すなわち2つの素数の積が2048bitなので、素数はおおむね 2^{1024} のオーダーになります。

ということで素数の数は。。。だいたい 2^{1014} (個)。

$$2^{1014} \times 1.4$$

個の
素数候補

- 300万（ 2^{21} くらい）個で太刀打ちできるレベルではない
 - 例えるならば、
1巻のヤムチャ v.s. セルの完全体

17

これだけの数の素数がある中、たった300万個の素数を作ったところでヒットするわけがないわけです。
宝くじに当たるよりもずっと低い確率なので実質「衝突」しない。

前座終わり

SSL証明書に親しみが湧いたところで

第二部

Certificate Transparencyに続きます

What is Certificate Transparency?



19

ようやく、今日のメインテーマです。

Certificate Transparencyとは何か？これは公開鍵証明書に入れるモノなので、証明書の見方が分かっているとチンプンカンプンになってしまう。

そのため、まずは素数をテーマにちょっと証明書鑑賞をしてみたわけです。

What is Certificate Transparency?

Certificate: 証明書
Transparency: 透明性

20

Certificateは証明書、そしてTransparencyは文脈に応じて訳すのが難しいですが、Googleは透明性と訳しています。
ここで言う透明性とは、「証明書の発行」という行為が透明であり、監査(Audit)可能であるということです。

なおこの頭文字を取って「CT」と呼びます、このスライドでも頻繁に出てきますので混乱しないでください。

What is Certificate Transparency?

Googleという悪の帝国が提唱している、
証明書発行の監視・監査の仕組み
(RFC 6962)

21

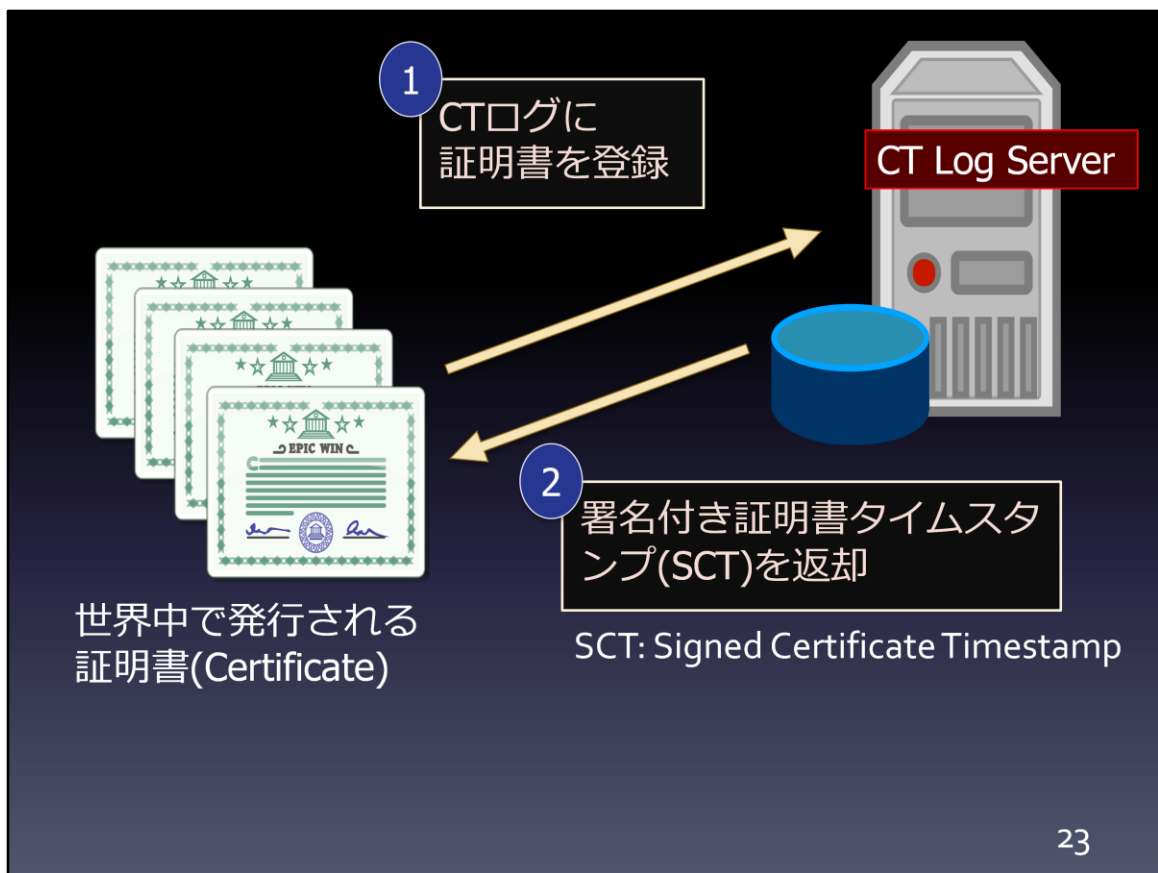
Certificate Transparencyは、皆さんご存じないでしょうがインターネットにはGoogleという悪の帝国がありまして。

CTはこのGoogleが提唱しており、SSL証明書の発行のログを誰でも参照可能な形で取ることで、監視(monitor)と監査(Audit)を可能にするものです。

既にRFC 6962(Experimental)が出ており、Google Chromeで対応しています。IEやSafariは当然まだ。



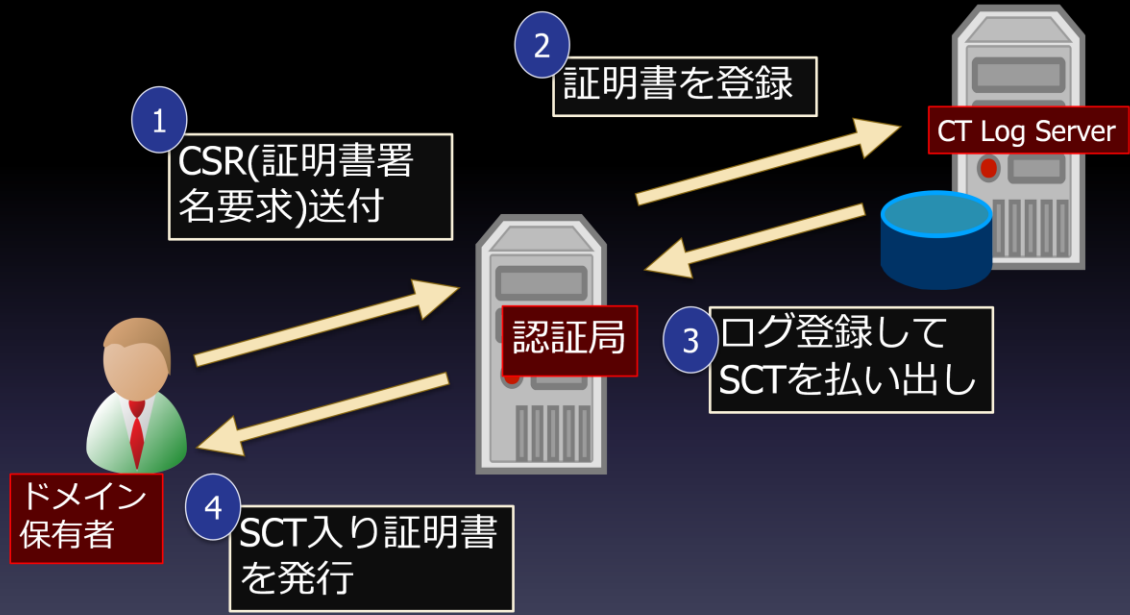
このように、ChromeでCTに対応している証明書を閲覧すると、「公開監査が可能です。」という表示と、「透明性に関する情報」というやつが出ます。



何がしたいのか。

究極の理想的には.....世界中で発行されるデジタル証明書を、すべてCTログに登録したいんです。CTログサーバは、証明書が登録されるとそのログを保存し、署名を付けたタイムスタンプを返します。これをSCT(Signed Certificate Timestamp)と言います。このSCT、今後何度も出てくるので覚えておいてください。証明書をCTログに登録時に発行される、タイムスタンプです。

※この図は不完全 (あとで解説)

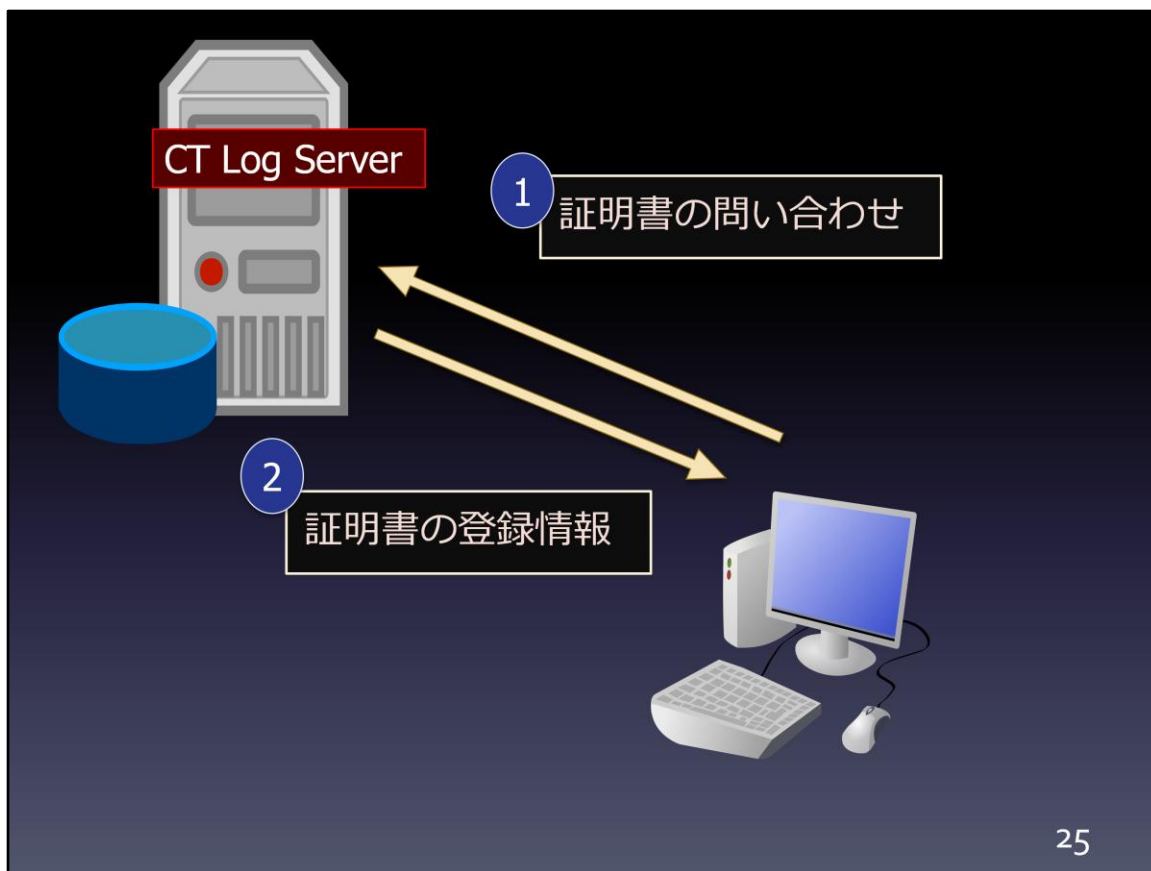


24

実際には、証明書はドメイン保有者が認証局に依頼するのが普通ですね。そのためこのような流れになります。

なお、この図は細かい部分をはしょっているので実は不正確です、それはあとで解説します。今は流れをまず理解するために簡略しています。

(具体的には、SCTを払い出してSCT入り証明書を作る部分が、Precertificateを作らないといけないのでこの図では正確ではない)



そして、CTログサーバは一般公開されているため誰でも参照可能です。
それがナニが嬉しいのかは、ちょっと後で解説します。



Certificate Transparencyですが、これはGoogleが提唱しているものなので、今のところ対応しているブラウザもChromeだけです。
それもMac版Chromeではダメで、今はWindows版Chromeしか対応していないようです。

Chromeには既にCTを確認する機能が実装されているので、簡単に確認できます。DigiCertなど、対応しているWebを見てみましょう。この通り、「公開監査が可能です」と出ています。また、「透明性に関する情報」から詳しいログ情報を見ることもできます。



先ほどの「透明性に関する情報」をクリックすると、このように監査ログが表示できます。

ここで出ているログ名「Google 'Pilot'」がCTログサーバのひとつです。



一方、これはCOOKPADの証明書ですが、監査ログが無いのでこのようなメッセージが出ます。「公開監査記録がありません。」とかなんだか気になる表示です。CTはGoogleが提唱して普及させようとしているものなので.....こういう不安を煽るようなメッセージは、わざとだと思います。悪の帝国のやり口ですね。

それにしてもCOOKPAD、言い方は悪いですがレシピサイトにEV証明書とは、随分まあ金持ちですなあ～。

```
echo Q | openssl s_client -connect www.digicert.ne.jp:443 | openssl x509 -text
```

..... (省略)

X509v3 extensions:

X509v3 Authority Key Identifier:

keyid:3D:D3:50:A5:D6:A0:AD:EE:F3:4A:60:0A:65:D4:F8:F8:D6:0F

**署名付き証明書タイムスタンプ
(Signed Certificate Timestamp; SCT)
OID=1.3.6.1.4.1.11129.2.4.2**

DNS:www.digicert.ne.jp, DNS:www.speedex.jp
DNS:digicert.ne.jp
X509v3 Key Usage: critical
Digital Signature, Key Encipherment
X509v3 Extended Key Usage:
TLS Web Server Authentication, TLS Web Client Authentication

..... (省略)

Authority Information Access:

OCSP - URI:http://ocsp.digicert.com

CA Issuers -

URI:http://ca.certs.digicert.com/DigiCertSHA2ExtendedCertificationServerCA.crt

constraints: critical

X509v3 Basic Constraints: critical
CA:FALSE

1.3.6.1.4.1.11129.2.4.2:

.z.x.v.....X.....gp

.....EOX.....GoE. N.....\$6&J

.b...g.....u...G.X..!...m.....a....u.bU...N=t.....>

..... (省略)

29

なおSCTは、先ほどOpenSSL手打ちで見たX509 Extensionsという領域に格納されています(実は規格上はそうではない形式のものもあるんですが、誰も使っていないので省略します)

opensslでも、フィールド名は出ませんが、このようにOIDそのままであればCTな証明書かは確認することができます。(最新のOpenSSL 1.0.2aなら対応しています)

OID=1.3.6.1.4.1.11129.2.4.2

ここに付いているのが、先ほどの図で証明書登録時にCTログサーバから払い出されたタイムスタンプ、SCT(Signed Certificate Timestamp)ですね。

Certificate Transparency, 何が嬉しい?

CTログサーバは誰でも閲覧できることから、多数の目による監視によって、不正な証明書発行を検知することができる（かもね）

ドメイン保有者は...	利用者(Webブラウザ)は...
自ドメインの証明書が勝手に発行されていないか、定期的にログをチェックすることで確認できる	接続先ホストが提示した証明書から、証明書発行時の監査ログを確認できる

30

Certificate Transparency, 何が嬉しいか。

CTログサーバは別にCA認証局などだけでなく、誰でも閲覧できます。皆さんも普通にURLを叩いてアクセスできますし、Webブラウザもアクセスします。

こうして多数の目による監視によって、不正な証明書発行、あるいは誤った証明書を検知したいというのが目的です。

例えば皆さんがmicrosoft.comのサーバ管理者だったとして、CTログサーバを定期的にチェックしてmicrosoft.comドメインへの証明書発行がされていないかを監視します。

ある日、自分は発行していないのにwww.microsoft.comの証明書が発行されたログがあれば、誰かが不正に証明書を発行した!?と検知できるわけです。

Certificate Transparency, (Googleには) 何が嬉しい?

- 世界中の証明書を手元に持つことができる
 - 証明書の種類、ドメイン名などのデータ
 - 認証局の証明書販売数の把握
- 認証局に対して、「上から目線」になれるカードを1枚持つことができる(SCTの発行)
- CTログサーバを、Googleが悪意を持ってアレをナニしないとは限らない

31

先ほどのが表向きの理由ですが。

おそらく悪の帝国Googleのメインの目的はこちらだと思います。何より、このようなログサーバを運営していると、世界中で発行されるSSL証明書が自動的に手元に集まってきます。

そのようにインターネットのトラフィックやリソースに関する大量のデータ収集というのは、あの悪の帝国が常々狙っているところです。

大量の証明書を収集することで、今後のデータ利用に繋げたいのでしょう。

また、もしGoogleが悪意を持ってログサーバに何かしらの細工をするかも.....ということも当然考えないといけません。

このようなことを考えると、例えばMacのSafariがCTに対応することはまず無いと思います (GoogleとAppleの仲の悪さは、まあなんとなく分かりますよね)

Chromeの横暴な計画

「Chromeでは、CTログに登録されていないEV証明書は、EVインジケータの表示をやめる」と宣言
(緑色の表示をしなくなると思われる)



Extended Validation in Chrome:

<http://www.certificate-transparency.org/ev-ct-plan>

32

またChromeの横暴で注意すべき点があります。

実はChromeは既に、CTログに対応していないEV証明書では緑表示をやめるということを宣言しています。(なんて身勝手な!)

銀行サイトなどでは、「安心のために緑表示を確認してください」という言い方をしていますが、あれができなくなってしまう。

ですからEV証明書を使う場合、CTログに対応させるか、あるいはChromeで緑表示されないことを無視するか、の2拓になります。

なお、Googleに証明書を提出して「どうかホワイトリストに載せてください」と頼むこともできます。現実的に、もう結構なサイトがこのホワイトリスト登録で対応しています。

気をつけろ! これが悪の帝国の兵器だ!



ひるめしを食べようと会社を出たところで、悪の帝国の車を見つけたので撮りました。
そのうち、私が写真を撮っている様子の写真が地図に載ります。

ここがイヤだよ Certificate Transparency



次にCertificate Transparencyのイヤなところも見ていきます。

ここがイヤだよ Certificate Transparency

CTログをクロールすることで、証明書に関する様々な情報を第三者が取得可能

35

私がイヤだな.....と思ったのはこれですね。

今回の発表に当たってCTログから色々データを取得してみたのですが、少しクロールしてみると、結構色んな情報が分かるなあ～、これあんまり出しちゃいけないんじゃないかなあ～、というのが率直な印象です。

こんなん見つけました：その1

- admin-test-aci1a-svls.bankofamerica.com
- admin-test-aci1b-svls.bankofamerica.com
- admin-test-dev1a-pipB.bankofamerica.com
- test.20150427.cloud.nifty.com
- *secret*.uccard.co.jp

外部非公開の、テスト・開発用と思われる
サーバのFQDNがモリモリ分かってしまう

36

例えば上の3つはBANK Of Americaのものですが、見るからにテスト用のサーバですよね。

4つめは、ニフティクラウドの何かですね。

あるドメインへの攻撃をする際、攻撃者はホスト列挙という行為を行うのですが、それが簡単にできます。

なお一番下のはUCカード、カード会社のものだったので.....たぶんテスト用サーバだろうなというFQDNでしたが、変なちょっかいを受けるとイヤなので、これは出さないことにしておきます。

こんな見つけました：Citrix XenApp

- citrix.●●●●.com
- citrix.■■■■.com
- citrix.▼▼▼▼.com
- citrix.○○○○.de
- など.....

maskしてます



他にも、こんな列挙ができました。Citrix XenAppという、デスクトップ仮想化のサーバが簡単に列挙できました。

CTにより、証明書を利用するサーバのFQDNを秘密にすることができなくなる

- We don't need metasploit anymore.
 - `msf > use auxiliary/gather/dns_enum`
- 関係者にしかFQDNを教えないことは、もちろん「気休め」でしか無い。でも気持ち悪い。
- この他、商品名などを入れたFQDNがサービスイン前に分かってしまう
 - (新サービス名?).(会社名).co.jp

38

というMetasploitを使わなくてもCTログから、攻撃対象ドメインのFQDN列挙が可能です。

また、FQDNから新サービスの名前が分かってしまうかもしれない。

認証局がCTログを(勝手に)登録していいの？

トレンドマイクロの主張

<http://esupport.trendmicro.com/solution/ja-JP/1106400.aspx> より引用

Certificate Transparency への対応について：

(略)また、あらゆるドメインに対して発行された証明書について、誰でも調べることができます。これは、SSL 証明書のセキュリティ上の問題にも見えますが、(中略) CT Log では、外部に公開されていない情報を閲覧することはできません。

内部サーバをご利用で、サーバの情報をCT Logに登録したくない場合は、OV 証明書を発行してください。現在CT Logに登録されるのはEV 証明書のみです。

39

さてこうなると、そもそも証明書を発行するときにそんな外部の第三者のサーバに登録しないで欲しい、というニーズも出てくるでしょう。

しかし、例えばトレンドマイクロのWebページにはこのように書かれています。

<http://esupport.trendmicro.com/solution/ja-JP/1106400.aspx>

なんと、FQDNの秘匿という観点では、EV証明書よりもOV証明書の方がセキュアです。逆転現象が起きています。

認証局がCTログを(勝手に)登録していいの？

トレンドマイクロの主張

<http://esupport.trendmicro.com/solution/ja-JP/1106400.aspx> より引用

Certificate Transparency への対応について：



CTログに登録されたく
なければ、EV証明書は諦め
てください。

バの情報をCT Log に登録したく
行してください。現在CT Log に登
録されるのはEV 証明書のみです。

40

なんだかぼかして書いていますが、要するにEV証明書をCTログに登録されることからは逃れられません。

既存の証明書もホワイトリストとして提出

トレンドマイクロ Q&A

<http://esupport.trendmicro.com/solution/ja-JP/1106400.aspx>

Trend Micro SSL の対応

Trend Micro SSL では、**2014 年 12 月 22 日**のシステム改修により、CT への対応を開始します。システム改修後は、新規に発行されるすべての EV 証明書に対し、CT Log から SCT が発行されるようになります。また、現在発行済みの EV 証明書については、ホワイトリストへ追加するため、Google 社へと提供します。

なお、Google 社は、各証明書に必要な SCT や CT Log サーバの数についても指定しており

キリッ

Trendmicro

現在発行済みのEV証明書については、
ホワイトリストへ追加するため、
Google社へと提供します。

41

さらに、Google(悪の帝国)は、CTログに登録せずに発行された証明書については、ホワイトリストを作っているから受け付けるよ! という声明を出しています。このホワイトリストに入れてもらわないと、EV証明書なのにChromeで緑色のインジケータが出ない、ということになります。これを重視して、一部の認証局はホワイトリスト登録を既にしてしまっています。

トレンドマイクロはこのように、ホワイトリストを提出済みです。一方、DigiCertは、「EV証明書を再発行すれば今はCTログに乗るから、そうしてくれ」というスタンスのようです。

Glimpse of Certificate Transparency Logs



42

次にCertificate Transparencyの見方について。

CT LogsはAPIなので、人間には見づらい

Retrieve Entries from Log:

<https://ct.googleapis.com/aviator/ct/v1/get-entries?start=1&end=1>

```
{"entries":[{"leaf_input":"AAAAAFAFBwW7swAAAVXMI  
IFUzCCBDugAwIBAgIHB6qchEONJzANBgkqhkiG9w0BAQ  
UFADCBYjELMAkGA1UEBhMCVVMxEDAQBgNVBAgTB0Fy  
aXpvcnRlcG9zaXRvcnkxMDAuBgNVBAMTZW50aW9uIE  
EBRMIMDc5NjkyODcwHhcNMjQwMzE1MTkwMDM5WjA/  
W4gQ29udHJvbCBWYWxpZC5NjcmVhbWlubWFTYXM  
FAAOCAQ8AMIIBCgKCAQEAXbEPu6OImDYgULJrWAF4e/2  
OVAbh8oEvg7cNZuwLZOk++Y4on79oUYvKVpB2M2qRka  
buG2+EyPipk7r0IAc8Grs1VCbgu1BLHIRynS7yYY3veUMZ  
Bd5AM+wUHyBBVYStJb914KFeZxNDw9nk/gTGGF/sEge  
mZoSb7+NytLNTsbU17JQRp601X6YS2TX3yAEohBPKqPLS
```

~~~~~  
> 突然のJSON <  
~~~~~  
Y^Y^Y^Y^Y^Y^Y

最初の方でCTログは誰でも参照可能と言いました。

<https://ct.googleapis.com/aviator/ct/v1/get-entries?start=1&end=1>

確かにそうなのですが、実はこれがAPIしか提供されておらず、人間が見るにはなかなかキツイ状態となっています。

作ってみた：Certificate Transparency ガチャ



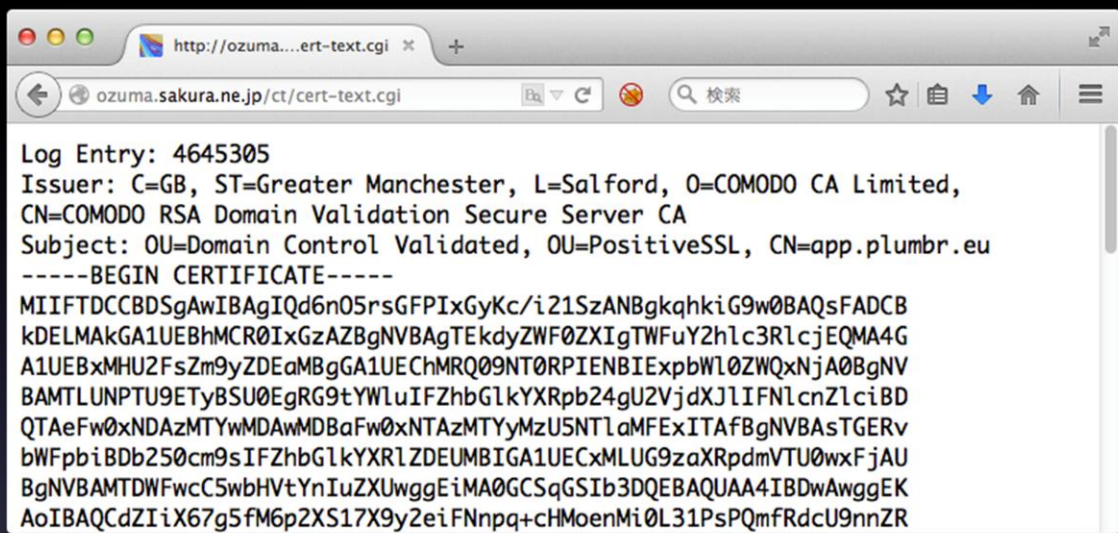
44

というわけで、作ってみました。簡単にCTログを見るためのツール、Certificate Transparencyガチャです。

<http://ozuma.sakura.ne.jp/ct/>

CTログには通し番号が振られているので、それを入れてもいいですし、下の「ガチャ」でランダムに引くこともできます。なんとこのガチャ、課金が不要です!!!!!!1

お手軽にCTログを覗くことが可能
(しかも無課金!!!!!!1)



The screenshot shows a web browser window with the address bar displaying `http://ozuma.sakura.ne.jp/ct/cert-text.cgi`. The page content shows a log entry for a certificate. The text is as follows:

```
Log Entry: 4645305
Issuer: C=GB, ST=Greater Manchester, L=Salford, O=COMODO CA Limited,
CN=COMODO RSA Domain Validation Secure Server CA
Subject: OU=Domain Control Validated, OU=PositiveSSL, CN=app.plumbr.eu
-----BEGIN CERTIFICATE-----
MIIFTDCCBDSgAwIBAgIQd6n05rsGFPIxGyKc/i21SzANBgkqhkiG9w0BAQsFADCB
kDELMakGA1UEBhMCR0IxGzAZBgNVBAGTEkdyZWZ0ZXIgdWVudXJlcnZlcnZlcnZl
A1UEBxMHU2FsZm9yZDEaMBGGA1UEChMRQ09NT0RPIENBIExpbWl0ZWQxNjA0BgNV
BAMTLUNPTU9ETyBSU0EgRG9tYWluIFZhbGlkYXRpb24gU2VjdXJlIFNlcnZlcnZl
QTAEFw0xNDZMTYwMDAwMDBaFw0xNTAzMTYyMzU5NTlaMFExITAfBgNVBAStGERv
bWVpbiBDb250cm9sIFZhbGlkYXRlZDEUMBIGA1UECXMUG9zaXRpdmVU0wxFjAU
BgNVBAMTDWw0cm9sIFZhbGlkYXRlZDEUMBIGA1UECXMUG9zaXRpdmVU0wxFjAU
AoIBAQcdZiix67g5fM6p2XS17X9y2eiFNnpq+cHMOenMi0L31PsPQmFRdcU9nnZR
```

<http://ozuma.sakura.ne.jp/ct/>

45

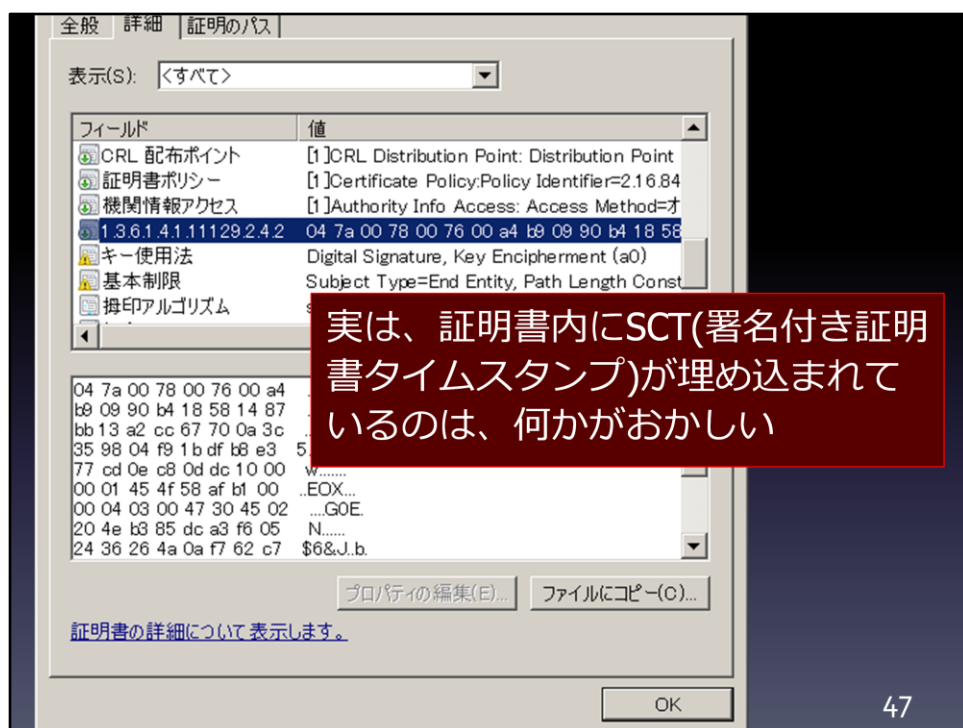
何度かやってみると、レアFQDNとか引けるかもしれません。なかなかヒマつぶしになります(自画自賛)。

<http://ozuma.sakura.ne.jp/ct/>

今日、話していないこと - SCTとPrecertificate



時間が無いので発表できなかったこと



先ほど何気なく、証明書に確かにSCT(覚えていますか? 要するにCTログから発行されたタイムスタンプです)が埋め込まれているのを見ましたが.....
鋭い人はもう気がついているでしょうが、これ、おかしい話なんです。

続きはブログで

- CT対応して証明書にSCTを入れるには、事前証明書(Precertificate)という気持ち悪いものを作らないといけない
- これはあまりスマートではない（と思う）
- 後日ブログに書こうと思います

48

CTログに、タイムスタンプ(SCT)をもらうにはまず証明書を作らないといけない。この時点では、当然SCTはまだ付いていない。
しかし皆さんがさっき見た証明書には、SCTが付いていた。なぜ？

実は証明書にSCTを入れるには、事前証明書(Precertificate)という気持ち悪いものを作らないといけないのです。
これは、後に発行する証明書と同じシリアルIDを持つなど、あまりスマートではない（と思います）。

後日ブログに書こうと思います

→書きました。

<http://d.hatena.ne.jp/ozuma/20150516/1431769141>

Certificate Transparency (個人的な感想です)



なんだか気持ちわるいお。。。
主要ブラウザには実装して欲しくないお。。。
でもFirefoxも対応するらしいお。。。

<https://threatpost.com/mozilla-to-support-certificate-transparency-in-firefox/109819>

49

FirefoxのCT対応

<https://threatpost.com/mozilla-to-support-certificate-transparency-in-firefox/109819>

デフォルトではONにしない、とは言っています。

参考

- GMO GlobalSignさんの記事が大変参考になりました。

- https://jp.globalsign.com/blog/2014/certificate_transparency.html

GMO GlobalSignさんの記事が大変参考になりました。

https://jp.globalsign.com/blog/2014/certificate_transparency.html

すみだセキュリティ勉強会

- <http://ozuma.sakura.ne.jp/sumida/>

51

すみだセキュリティ勉強会

<http://ozuma.sakura.ne.jp/sumida/>