Code Signing Certificate Misissuance in the Wild

Mason Hyman, Julia Steed

THE UNIVERSITY OF TENNESSEE
KNOXVILLE

Introduction and Motivation

What is the difference between code-signing and TLS certificates?

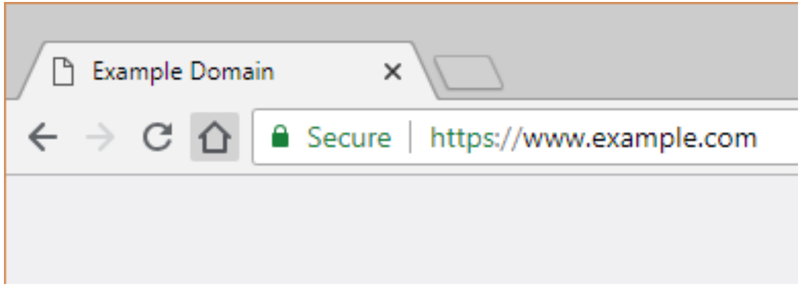
TLS Certificates:

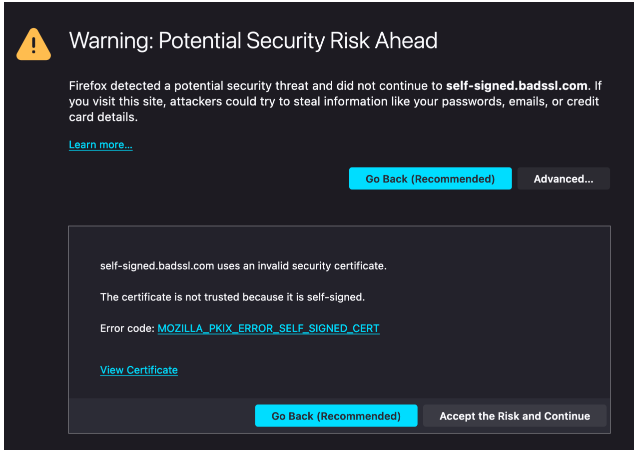
- Encrypts the data exchanged between a user's web browser and the web server.
- Provides confidentiality and integrity.
- Verifies the identity of the web server.

Code-Signing Certificates:

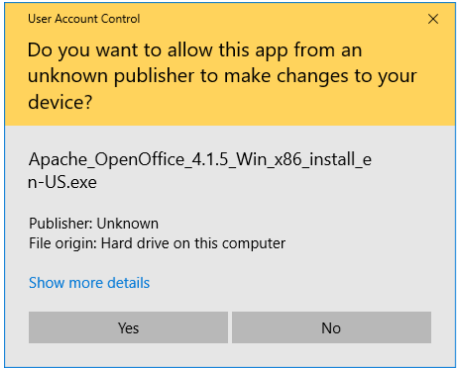
- Digitally signs software and executable code to improve the security of software distribution.
- Provides authenticity and integrity.
- Verifies the identity of the software publisher

TLS Certificate In Action:





Code Signing Certificate In Action:



Motivation

- Our goal with this project is to systematically analyze the errors made by CAs when issuing code-signing certificates.
- We define any direct violation of technical standards or community best practices as a **misissuance** by the issuing certificate authority.
- Our work is derived directly from *Tracking Certificate Misissuance in the Wild* (Kumar et. al.). This paper focused solely on TLS certificates; thus, we wish to extend their work.

Paper Source: https://zakird.com/papers/zlint.pdf

What are the standards?

- RFC5280 – Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile.
- Baseline Requirements for the Issuance and Management of Publicly-Trusted Code Signing Certificates (BRfCSC) – Issued Sep. 21, 2016

I E T F®

CAB CA/BROWSER FORUM

Baseline Requirements (cont.)

CAB CA/BROWSER FORUM

- The document walks through each section of an X.509 certificate profile and issues requirements for each field:

e. `keyUsage`

This extension MUST be present and MUST be marked critical.

The bit position for `digitalSignature` MUST be set. Bit positions for `keyCertSign` and `cRLSign` MUST NOT be set. All other bit positions SHOULD NOT be set.

7.1.4.2.2 Subject distinguished name fields - EV and Non-EV Code Signing Certificates

a. **Certificate Field:** `subject:commonName` (OID 2.5.4.3)
   Required/Optional: Required
   Contents: This field MUST contain the Subject's legal name as verified under Section 3.2.2 or 3.2.3.

7.1.3.2.1 RSA

The CA SHALL use one of the following signature algorithms:

- RSASSA-PKCS1-v1_5 with SHA-256
- RSASSA-PKCS1-v1_5 with SHA-384
- RSASSA-PKCS1-v1_5 with SHA-512
- RSASSA-PSS with SHA-256
- RSASSA-PSS with SHA-384
- RSASSA-PSS with SHA-512

In addition, the CA MAY use `RSASSA-PKCS1-v1_5 with SHA-1` if one of the following conditions are met:

- It is used within Timestamp Authority Certificate and the date of the `notBefore` field is not greater than 2022-04-30; or,
- It is used within an OCSP response; or,
- It is used within a CRL; or,
- It is used within a Timestamp Token and the date of the `genTime` field is not greater than 2022-04-30.

Methods

Data Collection

- To analyze the misissuance of code-signing certificates, we needed a large corpus of signed binaries.
- The most commonly available source of signed binaries is from antivirus and security research companies as they have already collected malware samples.
- For this project, we collected datasets from popular antivirus vendors and open source repositories of malicious binaries. From these binaries, we extracted the code signing certificates.

	Certificates:	Issued after 9/21/16:
Malcert Dataset:	6,322	0
Symantec Dataset:	132,485	10,270
Sorel Dataset:	22,559	7,968
VirusShare Dataset:	20,121	1,561
	Total:	19,799

Data Processing

- To process a large volume of certificates, we forked **ZLint**, the X.509 certificate linter used by Kumar et. al.
- Because the requirements for TLS certificates and Code-signing certificates are different, we had to make many modifications to the original ZLint program.

Lints Added: 96

Requirement Coverage (Profile-only): ~97%

How does ZLint work?

- A linter is a static code analysis tool typically used to flag programming errors, bugs, or syntactic errors.
- In ZLint's case, the program checks for misissuances in X.509 certificates by comparing individual fields to their expected or required value.

```
masonh@moa6:/datasets/masonh$ openssl x509 -in data/symantec/cs_cert_0001753ab039953627e02e1d2a8f8440.pem -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            07:16:5f:53:78:f7:a5:d0:ab:f8:b9:48:87:f1:60:0c
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C = US, O = "VeriSign, Inc.", OU = VeriSign Trust Network, OU = Terms of use at https://www.veri
        Validity
            Not Before: Nov 21 00:00:00 2013 GMT
            Not After : Dec  4 23:59:59 2014 GMT
        Subject: C = JP, ST = Tokyo, L = Minato-ku, O = "JAL Information Technology Co.,Ltd.", OU = Digital ID C
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:8d:3b:ec:bb:7f:d6:5c:93:32:d8:24:88:6d:f5:
                    9e:3b:3a:da:d3:32:cb:30:1b:6d:7c:35:a9:a9:b1:
                    23:e1:70:4a:1e:72:88:60:6b:39:03:d7:ed:be:36:
                    09:f1:2a:ac:99:b6:5b:65:23:1c:06:37:53:1f:ad:
                    35:d1:d3:aa:45:a7:42:38:0a:8b:a6:38:87:aa:da:
                    0a:98:a5:a2:c1:7f:0d:10:13:f9:bc:fe:38:4e:c1:
                    a5:65:2a:8c:77:27:87:d5:19:9f:b5:8c:7e:52:96:
                    4a:e7:5e:45:82:9f:d3:82:5a:f6:b6:dc:9b:b1:39:
                    16:b4:39:c4:ec:0f:73:83:44:71:c4:d1:30:f0:ae:
                    c1:ba:fc:b8:1f:31:2d:09:93:aa:9f:88:dd:e5:2b:
                    7a:b0:43:83:cb:8c:72:c0:2b:65:c5:f3:23:ff:10:
                    38:d1:79:81:19:de:2b:89:58:4e:ab:99:cf:80:3a:
                    cd:af:1b:2d:c5:e5:81:2e:d2:62:e9:c1:3e:d0:87:
                    b2:c6:77:98:97:d1:43:aa:e9:af:c8:55:ff:86:75:
                    1b:d1:86:9b:ff:fa:c4:2d:9d:79:5b:f4:3d:b5:07:
                    95:87:8c:a4:2d:65:85:07:8c:0c:42:61:2c:f8:8c:
                    fb:d6:a8:a2:4f:5b:03:81:2b:98:4d:5c:81:b1:dd:
                    4d:83
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Key Usage: critical
                Digital Signature
            X509v3 CRL Distribution Points:
                Full Name:
                  URI:http://csc3-2010-crl.verisign.com/CSC3-2010.crl
            X509v3 Certificate Policies:
                Policy: 2.16.840.1.113733.1.7.23.3
                  CPS: https://www.verisign.com/rpa
            X509v3 Extended Key Usage:
                Code Signing
            Authority Information Access:
                OCSP - URI:http://ocsp.verisign.com
                CA Issuers - URI:http://csc3-2010-aia.verisign.com/CSC3-2010.cer
            X509v3 Authority Key Identifier:
                CF:99:A9:EA:7B:26:F4:4B:C9:8E:8F:D7:F0:05:26:EF:E3:D2:A7:9D
```

How does ZLint work? (cont.)

- In our testing, we measured that ZLint could perform at roughly 250,000 certificates per hour.
- ZLint is written in Golang and is configured to write its results to a sqlite3 database.

```go
func (l *certExtensionsVersionNot3) Execute(cert *x509.Certificate) *LintResult {
    /*
     *  Check if the cert vesion is 3. Note this value is not zero index as specified in RFC 2459
     */
    if cert.Version != 3 {
        return &LintResult{Status: Error}
    }
    return &LintResult{Status: Pass}
}
```

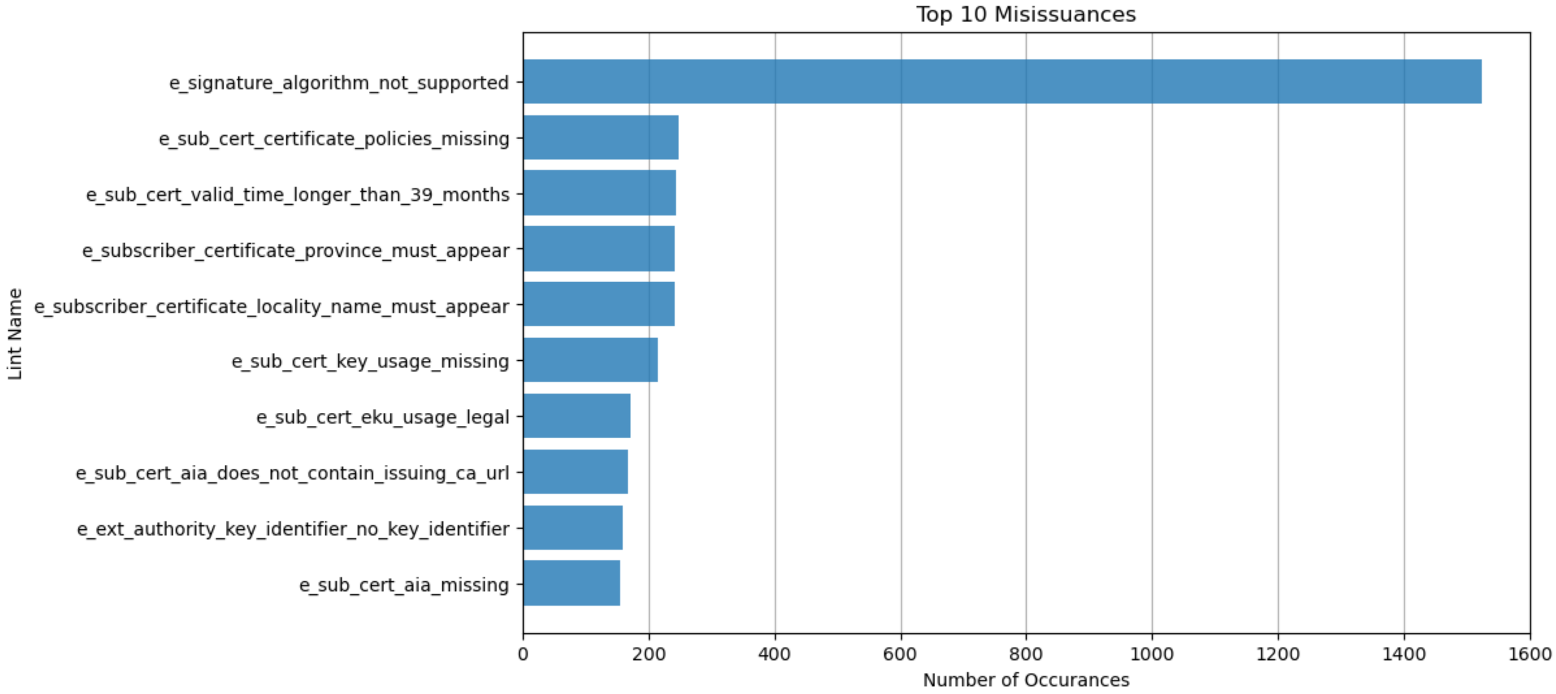
Example: Execute function for Lint that checks certificate version is 3

Results

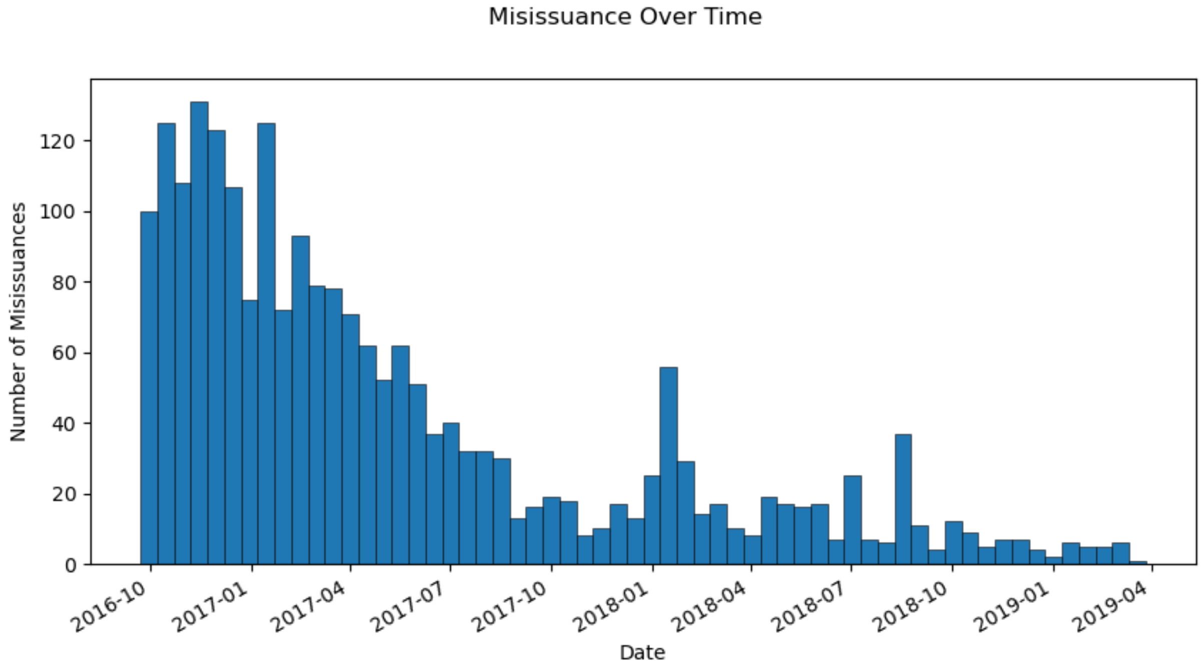
Misissuance Quantified

Dataset:	# of Unique Certs Misissued	% of Total	# of Passed	# of Errors	# of Warnings
Symantec	1,248	12.1512%	341,599	548	913
Sorel	706	8.8604%	186,758	2,040	476
VirusShare	237	16.3997%	52,370	74	182
All	**2,192**	**11.0713%**	**580,727**	**2,662**	**1,571**

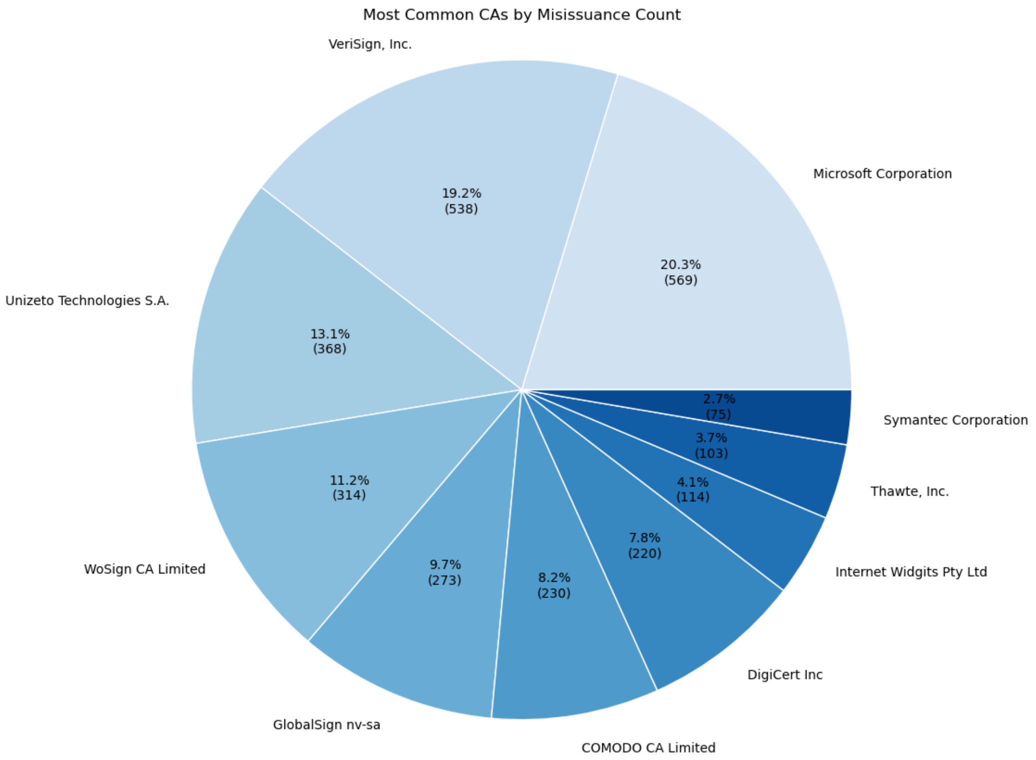
Most Common Misissuances



Misissuance Over Time



Most Common CAs to Misissue

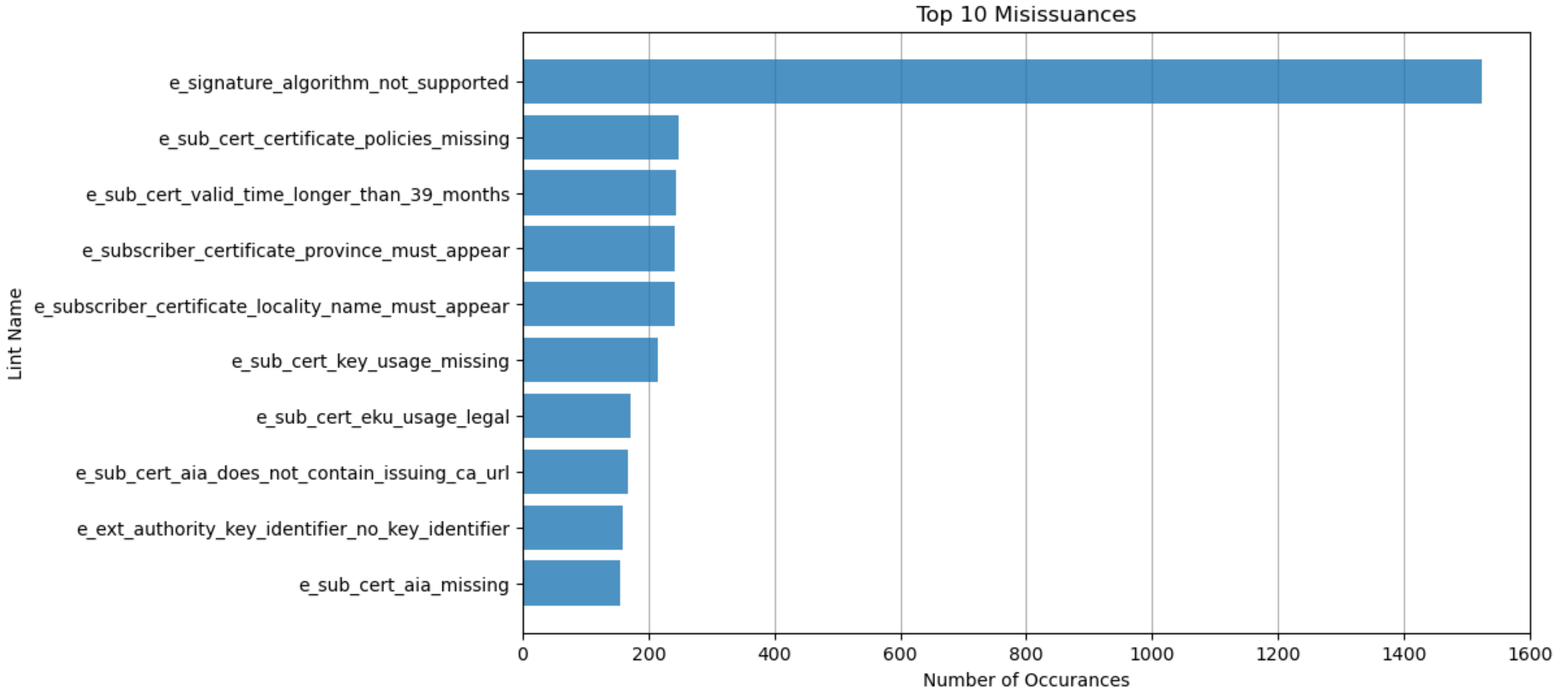


Discussion

So what do the results tell us?

- Based on the identification that **11.1%** of code-signing certificates contained a violation of accepted baselines and best practices, there is potential for improvement by certificate authorities.
- Compared to the conclusion that only 3.3% of TLS certificates contained violations made by Kumar et. al, one may generalize that certificate authorities make more errors when issuing code-signing certificates.

Most Common Misissuances



What do the most common misissuances tell us?

- By a margin of over 1,000 occurrences, <u>signature algorithm misissuances</u> were the most commonly seen among the dataset.
- This is mostly due to the transition between RSA-2048 and RSA-3072 and the unclear required transition dates issued by the CA/B Forum.
- It should be noted that CAs should be less to blame for these misissuances as the responsibility to set clear transition dates between requirements falls upon the CA/B Forum.

What do the most common misissuances tell us? (cont.)

- In the case of key usage and extended key usage misissuances, CAs are clearly to be blamed as the requirements are very straightforward.
- In 215 cases (1.11%), the X.509 key usage field was not included despite being required as a critical field.
- When not included, certificates become vulnerable to misuse by the holder. If stolen, malicious actors may use the certificate freely to perform unintended actions.

What do the most common misissuances tell us? (cont.)

- Within the most common misissuances, there were 476 instances of Authority Information Access (AIA) violations.
- This field is required to contain information and links that browsers and other applications can use to check the validity and revocation status of certificates.
- Without a proper AIA field, certificate revocation is rendered ineffective. This is a large vulnerability in the current PKI landscape.

Next Steps

- In order to get a full picture of misissuance as it relates to code-signing certificates, many more certificates are needed.
- Ideally, a majority of these certificates would come from benign binaries, which we did not have in this project.
- A large corpus of benign, signed binaries would help to rule out some of the noise inherent to datasets of malicious binaries.
- It should be noted that collected a large supply of benign binaries is both difficult and computationally expensive (bandwidth, storage).

Next Steps (cont.)

- With more certificates, one may begin to examine the differences between the misissuance of Extended-Validation (EV) and Non-EV code-signing certificates.
- By examining EV certificates, there may be better indicators as to the health of the code signing ecosystem.
- Additionally, a deep dive into the effectiveness of the current baseline requirements should be performed as well as examining their usability by CAs.

Code Signing Certificate Misissuance in the Wild

Mason Hyman, Julia Steed

THE UNIVERSITY OF
TENNESSEE
KNOXVILLE