Understanding TLS certs validation errors

A talk on *usable* security...

Summary of 2018 research by Martin Ukrop, Lydia Kraus and Vashek Matyas

*DevConf 2019, 26. 1. 2019*

**Martin Ukrop**, mukrop@mail.muni.cz

**Masaryk University**, CRoCS

Ph.D. research cooperation with Red Hat Czech
8:03 AM

Emergency Alert
BALLISTIC MISSILE THREAT INBOUND TO HAWAI. SEEK IMMEDIATE SHELTER. THIS IS NOT A DRILL.

8:03

The phone beeps.
A text comes.
38 minutes pass...
MISSILE ALERT IN ERROR THERE IS NO THREAT
Cause? Bad warning system UI!

1. State EOC

- PACOM (CDW) - STATE ONLY
- BMD False Alarm
- Amber Alert (CAE) - Kauai County Only
- Amber Alert (CAE) Statewide
- 1. TEST Message
- PACOM (CDW) - STATE ONLY
- Tsunami Warning (CEM) - STATE ONLY
- DRILL-PACOM (DEMO) STATE ONLY
- Landslide - Hana Road Closure
- Amber Alert DEMO TEST
- High Surf Warning North Shores
What about…

Encrypted email? Sure!

Yes.
Bad user interface.
Noting to to with security.
Usability of PGP 5.0 (1999)

Why Johnny Can’t Encrypt

A Usability Evaluation of PGP 5.0

ALMA WHITTEN AND J. D. TYGAR

User errors cause or contribute to most computer security failures, yet user interfaces for security still tend to be clumsy, confusing, or near nonexistent. Is this simply because of a failure to apply standard user interface design techniques to security? We argue that, on the contrary, effective security requires a different usability standard, and that it will not be achieved through the user interface design techniques appropriate to other types of consumer software.

To test this hypothesis, we performed a case study of a security program that does have a good user interface by general standards: PGP 5.0. Our case study used a cognitive walkthrough analysis together with a laboratory user test to evaluate whether PGP 5.0 can be used successfully by cryptography novices to achieve effective electronic mail security. The analysis found a number of user interface design flaws that may contribute to security failures, and the user test demonstrated that when our test participants were given 90 minutes in which to sign and encrypt a message using PGP 5.0, the majority of...
Has the world moved on? (Microsoft Office + PGP 9, 2006)

Why Johnny Still Can’t Encrypt:
Evaluating the Usability of Email Encryption Software

Steve Sheng
Engineering and Public Policy
Carnegie Mellon University
shengx@cmu.edu

Levi Broderick
Electrical and Computer Engineering
Carnegie Mellon University
lpb@ece.cmu.edu

Colleen Alison Koranda
HCI Institute
Carnegie Mellon University
ckoranda@andrew.cmu.edu

Jeremy J. Hyland
Heinz School of Public Policy and Management
Carnegie Mellon University
jhyland@andrew.cmu.edu

ABSTRACT
Our research seeks to understand the current usability situation of email encryption software, particularly PGP 9 in comparison to previous studies of PGP 5. We designed a pilot study to find current problems in the following areas: create a key pair, get public keys, verify public keys, encrypt an email, sign an email, decrypt an email, verify a digital signature, and save a backup of public and private keys.

1. INTRODUCTION
In the seminal paper “Why Johnny Can’t Encrypt”, Whitten and Tygar [1] showed that users have great difficulty using email encryption software PGP. In the study, only 4 out of 12 participants were able to correctly sign and encrypt an email message to test user’s response to PGP’s automatic decryption.

2. MAJOR FINDINGS
2.1 Verify Keys
We found that key verification and signing is still severely lacking, such that no user was able to successfully verify their keys. Similar to PGP 5, users had difficulty with signing keys. Three of our users were not able to verify the validity of the key successfully and did not understand the reasoning to do so. Four users were not able to sign the key, these users attempted to but struggled with the interface. They did not understand that in order to ‘verify,’ they must ‘sign’ the key rather than just click ‘verify.’
And now? Please? (Mailvelope, 2015)

Why Johnny Still, Still Can’t Encrypt:
Evaluating the Usability of a Modern PGP Client

Scott Ruoti, Jeff Andersen, Daniel Zappala, Kent Seamons
Brigham Young University
{ruoti, andersen} @ isrl.byu.edu, {zappala, seamons} @ cs.byu.edu

ABSTRACT
This paper presents the results of a laboratory study involving Mailvelope, a modern PGP client that integrates tightly with existing webmail providers. In our study, we brought in pairs of participants and had them attempt to use Mailvelope to communicate with each other. Our results shown that more than a decade and a half after Why Johnny Can’t Encrypt, modern PGP tools are still unsuitable for the masses. We finish with a discussion of pain points encountered using Mailvelope, and discuss what might be done to address them in future PGP systems.

Author Keywords
Security, usability, secure email, PGP

ACM Classification Keywords
H.1.2. Models and Principles: User/Machine Systems—human factors; H.5.2. Information Interfaces and Presentation (e.g. HCI): User Interfaces—user-centered design

INTRODUCTION
Usable, secure email is still an open problem more than 15 years after it was first studied by Whitten et al. [10]. Six years after the original Johnny paper, Sheng et al. showed that PGP 9 was still difficult for users to operate correctly [9]. In this paper, we attempt to see if in the last decade, modern PGP-based tools have improved to the point where users can.

In our study of 20 participants, grouped into 10 pairs of participants who attempted to exchange encrypted email, only one pair was able to successfully complete the assigned tasks using Mailvelope. All other participants were unable to complete the assigned task in the one hour allotted to the study. This demonstrates that encrypting email with PGP, as implemented in Mailvelope, is still unsuitable for the masses.

Our results also shed light on several ways that PGP-based tools could be improved. First, integrated tutorials would be helpful in assisting first time users in knowing what they should be doing at any given point in time. Second, an approachable description of public key cryptography could help users correctly manage their own keys. Third, in line with previous work by Atwater et al. [1], we find that PGP-based tools would be well served by offering automatically generated emails for unknown recipients asking them to install the PGP software, generate a public key, and share it with the sender. Finally, the PGP block itself could be enhanced to help non-PGP users who receive an encrypted email know how to work with their friend to get an encrypted message they will be able to read.

RELATED WORK
Whitten and Tygar [10] conducted the first formal user study of a secure email system (i.e., PGP 5), uncovering serious usability issues with key management and users' understanding.
It’s not just academia...

Because of popular demand, here’s the collection of reasons to prefer more advanced cryptographic communications tools and stop investing in the old PGP over e-mail architecture, the problem mostly being e-mail rather than PGP.

Pretty Good Privacy is better than no encryption at all, and being end-to-end is also better than relying on SATP over TLS (that is, point-to-point) between the mail servers while the message is unencrypted in-between, but is it still a good choice for the future? Is it something we should recommend to people who are asking for better privacy today?

The text concludes mentioning some of the existing alternatives, so, against my recommendation this is not about not using encryption. It is about not falling into the intellectual trap of giving backwards compatibility the highest priority.

1. Downgrade Attack: The risk of using it wrong.
Let’s start validating TLS certificates...

But surely, it’s only end users, isn’t it?
Oh, I need to validate this certificate...

[attendee@devconf ~]$ openssl verify cert-chain.pem
Oh, I need to validate this certificate...

[attendee@devconf ~]$ openssl verify cert-chain.pem

CN = secret.devconf.cz, O = Red Hat, Inc., C = CZ
error 47 at 0 depth lookup:
    permitted subtree violation
error cert-chain.pem: verification failed
Solution 1:

The user should know!

Not My Problem
Solution 2:
Investigate!
Understand!
Decide!

Hostname validation

OpenSSL 1.1.0 provides built-in functionality for hostname checking and validation. Viktor Dukhniv provided this answer on Stack Overflow in January 2015. It's been available in Master since that time. The code is beginning to see widespread testing as the 1.1.0 approaches.

One common mistake made by users of OpenSSL is to assume that OpenSSL will validate the hostname in the certificate during certificate validation. This is not necessary for certificate validation. The hostnames in the certificate are validated separately, during hostname checking.

A man page on hostname validation has been available since 1.0.2. Also see the `X509_check_host()` function.

Example Usage

The following is from the OpenSSL manual and shows how you could use OpenSSL's built-in hostname validation:

```c
const char servername[] = "www.example.com";
SSL *ssl = NULL;
X509_VERIFY_PARAM *param = NULL;
...

servernames = "www.example.com";
ssl = SSL_CTX_new(ssl_version);
ssl = SSL_CTX_use_certificate_file(ssl, certificate, SSL_FILETYPE_PATH);
ssl = SSL_CTX_use_PrivateKey_file(ssl, private_key, SSL_FILETYPE_PEM);
/* Enable automatic hostname checks */
X509_VERIFY_PARAM_set1_hostflags(param, X509_CHECK_FLAG_NO_PARTIAL_HILDCARDS);
if (X509_VERIFY_PARAM_set1_host(param, servername, sizeof(servername) - 1)) {
    return 0;
}
/* Enable peer verification, (with a non-null callback if desired) */
```
But there are MANY possible errors...

[attendee@devconf ~]$ man openssl verify | grep ...

X509_V_OK, X509_V_ERR_UNSPECIFIED, X509_V_ERR_UNABLE_TO_GET_ISSUER_CERT, X509_V_ERR_UNABLE_TO_GET_CRL,
X509_V_ERR_UNABLE_TO_DECRYPT_CERT_SIGNATURE, X509_V_ERR_UNABLE_TO_DECRYPT_CRL_SIGNATURE,
X509_V_ERR_UNABLE_TO_DECODE_ISSUER_PUBLIC_KEY, X509_V_ERR_CERT_SIGNATURE_FAILURE,
X509_V_ERR_CRL_SIGNATURE_FAILURE, X509_V_ERR_CERT_NOT_YET_VALID, X509_V_ERR_CERT_HAS_EXPIRED,
X509_V_ERR_CRL_NOT_YET_VALID, X509_V_ERR_CRL_HAS_EXPIRED, X509_V_ERR_ERROR_IN_CERT_NOT_BEFORE_FIELD,
X509_V_ERR_ERROR_IN_CERT_NOT_AFTER_FIELD, X509_V_ERR_ERROR_IN_CRL_LAST_UPDATE_FIELD,
X509_V_ERR_ERROR_IN_CRL_NEXT_UPDATE_FIELD, X509_V_ERR_OUT_OF_MEM, X509_V_ERR_DEPTH_ZERO_SELF_SIGNED_CERT,
X509_V_ERR_SELF_SIGNED_CERT_IN_CHAIN, X509_V_ERR_UNABLE_TO_GET_ISSUER_CERT_LOCALLY,
X509_V_ERR_UNABLE_TO_VERIFY_LEAF_SIGNATURE, X509_V_ERR_CERT_CHAIN_TOO_LONG, X509_V_ERR_CERT_REVOKED,
X509_V_ERR_INVALID_CA, X509_V_ERR_PATH_LENGTH_EXCEEDED, X509_V_ERR_INVALID_PURPOSE,
X509_V_ERR_CERT_UNTRUSTED, X509_V_ERR_CERT_REJECTED, X509_V_ERR_SUBJECT_ISSUER_MISMATCH,
X509_V_ERR_AKID_SKID_MISMATCH, X509_V_ERR_AKID_ISSUER_SERIAL_MISMATCH, X509_V_ERR_KEYUSAGE_NO_CERTSIGN,
X509_V_ERR_UNABLE_TO_GET_CRL_ISSUER, X509_V_ERR_UNHANDLED_CRITICAL_EXTENSION,
X509_V_ERR_KEYUSAGE_NO_CRL_SIGN, X509_V_ERR_UNHANDLED_CRITICAL_CRL_EXTENSION, X509_V_ERR_INVALID_NON_CA,
X509_V_ERR_PROXY_PATH_LENGTH_EXCEEDED, X509_V_ERR_PROXY_SUBJECT_INVALID,
X509_V_ERR_KEYUSAGE_NO_DIGITAL_SIGNATURE, X509_V_ERR_PROXY_CERTIFICATES_NOT_ALLOWED,
X509_V_ERR_INVALID_EXTENSION, X509_V_ERR_INVALID_POLICY_EXTENSION, X509_V_ERR_NO_EXPLICIT_POLICY,
X509_V_ERR_DIFFERENT_CRL_SCOPE, X509_V_ERR_UNSUPPORTED_EXTENSION_FEATURE, X509_V_ERR_UNNESTED_RESOURCE,
X509_V_ERR_PERMITTED_VIOLATION, X509_V_ERR_EXCLUDED_VIOLATION, X509_V_ERR_SUBTREE_MINMAX,
X509_V_ERR_APPLICATION_VERIFICATION, X509_V_ERR_UNSUPPORTED_CONSTRAINT_TYPE,
X509_V_ERR_UNSUPPORTED_CONSTRAINT_SYNTAX, X509_V_ERR_UNSUPPORTED_NAME_SYNTAX,
X509_V_ERR_CRL_PATH_VALIDATION_ERROR, X509_V_ERR_PATH_LOOP, X509_V_ERR_SUITE_B_INVALID_VERSION,
X509_V_ERR_SUITE_B_INVALID_ALGORITHM, X509_V_ERR_SUITE_B_INVALID_CURVE,
X509_V_ERR_SUITE_B_INVALID_SIGNATURE_ALGORITHM, X509_V_ERR_SUITE_B_LOS_NOT_ALLOWED,
X509_V_ERR_SUITE_B_CANNOT_SIGN_P_384_WITH_P_256, X509_V_ERR_HOSTNAME_MISMATCH, X509_V_ERR_EMAIL_MISMATCH,
X509_V_ERR_IP_ADDRESS_MISMATCH, X509_V_ERR_DANE_NO_MATCH, X509_V_ERR_EE_KEY_TOO_SMALL,
X509_V_ERR_CA_KEY_TOO_SMALL, X509_V_ERR_CA_MD_TOO_WEAK, X509_V_ERR_INVALID_CALL, X509_V_ERR_STORE_LOOKUP,
X509_V_ERR_NO_VALID_SCTS, X509_V_ERR_PROXY_SUBJECT_NAME_VIOLATION, X509_V_ERR_OCSP_VERIFY_NEEDED,
X509_V_ERR_OCSP_VERIFY_FAILED, X509_V_ERR_OCSP_CERT_UNKNOWN.
Problem statement

- How do people in IT perceive certificate flaws?
  - Do they understand the cause?
  - Do they see the (security) consequences?
  - Further complication: Sometimes deliberate deployment of invalid TLS certificates...
Problem statement

• How do people in IT perceive certificate flaws?
  – Do they understand the cause?
  – Do they see the (security) consequences?
  – Further complication: Sometimes deliberate deployment of invalid TLS certificates...

• How do error messages help comprehension?
  – Do they matter much? Can they be better?
Task: You’d LOVE to register via Google...
Open source! Let’s write a patch!

[attendee@devconf ~]$ ./testConnection server_google
Chosen provider: Google
Connecting to accounts.google.com...
Connection success.
Certificate chain saved to server_google.pem.

Certificate validation failed!
  Permitted subtree violation
  (X509_V_ERR_PERMITTED_VIOLATION)
Task procedure (simplified)

1. Try to understand the problem and risks. (Do whatever would you do.)
2. How much do you trust the server having this certificate?
3. Later: Describe in your own words what was the problem with the certificate.
So we discussed it with 75 developers...
Participant stats

• 75 participants (👥)
  – 67 with recorded interviews
• 95% employed in IT (median 8 years)
• 67% have formal education in IT
• 91% used OpenSSL before
  – 25% NSS, 25% Java Keytool, 19% GnuTLS
Results I.

What perceptions do people in IT have? (w.r.t. cert flaws)
Case 1: **OK** (Github)

[attendee@devconf ~]$ ./testConnection server_github
Chosen provider: Google
Connecting to auth.github.com...
Connection success.
Certificate chain saved to server_github.pem.

Certificate validation return value:
  ok (X509_OK)
Case 1: OK (Github)

61 �GORITHM NoIssue*

NoIssue:

“There wasn’t a problem, it was good, OK.” [P22]
Case 1: OK (Github)

61 ♗ Nolssue*
13 ♗ ExtraCheck

ExtraCheck:
“\(I\ think\ it\ was\ safe,\ but\ I\ looked\ into\ the\ cert\ and\ I\ couldn’t\ find\ anything\ wrong,\ so\ I\ would\ trust\ it...\)"

[P13]
Case 1: OK (Github)

61 ♙ NoIssue*
13 ♙ ExtraCheck
12 ♙ BugFree

BugFree: “[...] everything looked fine and I thought: ‘Well, if the testing tool is good, I’ll trust that.’” [P77]
Case 2: Expired (Microsoft)

[attendee@devconf ~/.]$ ./testConnection server_microsoft
Chosen provider: Microsoft
Connecting to login.microsoft.com...
Connection success.
Certificate chain saved to server_microsoft.pem.

Certificate validation return value:
certificate has expired
(X509_V_ERR_CERT_HAS_EXPIRED)
Case 2: Expired (Microsoft)

NoLonger:

“Microsoft certificate has expired, it’s out of date.”

[P30]
Case 2: Expired (Microsoft)

Mistake:

“[I have] some feeling like maybe it could be just forgotten and they’re about to do it, they’re about to renew it or something.”

[P10]
Case 2: Expired (Microsoft)

Common:

“Ah, right, so, expired certificates are pretty common, so from what I can see [...]” [P01]
Case 2: Expired (Microsoft)

62 ♙ NoLonger*
27 ♙ Mistake
18 ♙ Common
14 ♙ OKBefore*

OKBefore:

“So it was valid in the past, and I looked at the date [...]” [P18]
Case 2: Expired (Microsoft)

62 ☐ No Longer*
27 ☐ Mistake
18 ☐ Common
14 ☐ OK Before*
13 ☐ Reputation

Reputation:

“If it’s like a small businesses from my local neighborhood, I would probably trust them.”

[P62]
Case 2: Expired (Microsoft)

62 ♙ NoLonger*
27 ♙ Mistake
18 ♙ Common
14 ♙ OKBefore*
13 ♙ Reputation
8 ♙ Attack

Attack:
“[...] it cannot be predicted if the attacker has stolen a certificate which was previously valid and has been revoked or [...]” [P37]
Case 3: Self-signed (Fedora project)

[attendee@devconf ~]$ ./testConnection server_fedora
Chosen provider: Fedora Project
Connecting to id.fedoraproject.com...
Connection success.
Certificate chain saved to server_fedora.pem.

Certificate validation return value:
  self signed certificate
  (X509_V_ERR_DEPTH_ZERO_SELF_SIGNED_CERT)
Case 3: Self-signed (Fedora project)

ByItself:

“That it is not signed by the other authority, but it’s signed by itself.” [P15]
Case 3: Self-signed (Fedora project)

NoCA:

“It means that it was signed by local server for which it was generated. It was not signed by official authority.” [P20]
Case 3: Self-signed (Fedora project)

AnyoneCan:

“Self-signed certificate? Anyone can create self-signed certificates.”

[P78]
Case 3: Self-signed (Fedora project)

IfExpected:

“If I knew that the certificate should be self-signed, I could consider it trustworthy.”

[P09]
Case 3: Self-signed (Fedora project)

50  ByItself*
28  NoCA*
21  AnyoneCan*
10  IfExpected
10  Internal

Internal:

“[…] and it’s usually used either by internally or for testing purposes. It shouldn’t be used publicly.” [P11]
### Case 3: Self-signed (Fedora project)

<table>
<thead>
<tr>
<th>Count</th>
<th>Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>ByItself*</td>
</tr>
<tr>
<td>28</td>
<td>NoCA*</td>
</tr>
<tr>
<td>21</td>
<td>AnyoneCan*</td>
</tr>
<tr>
<td>10</td>
<td>IfExpected</td>
</tr>
<tr>
<td>10</td>
<td>Internal</td>
</tr>
<tr>
<td>8</td>
<td>Attack</td>
</tr>
</tbody>
</table>

**Attack:**

“[…] because that can be any hacker, [they] can phish that and malware can be added.” [P66]
HALF WAY POINT
Case 4: Hostname mismatch (Facebook)

[attendee@devconf ~]$ ./testConnection server_facebook
Chosen provider: Facebook
Connecting to oauth.facebook.com...
Connection success.
Certificate chain saved to server_facebook.pem.

Certificate validation return value:
Hostname mismatch
(X509_V_ERR_HOSTNAME_MISMATCH)
Case 4: Hostname mismatch (Facebook)

BadName:

“The last one server, Facebook, [the certificate] was issued for a different hostname.” [P39]
Case 4: Hostname mismatch (Facebook)

NameCheck:

“[…] because it is not Facebook, it is Facesbook or something like that.”

[P57]
Case 4: Hostname mismatch (Facebook)

50 ♙ BadName*
27 ♙ NameCheck
22 ♙ Attack

Attack:

“It can be some phishing site or something like this.”

[P76]
Case 4: Hostname mismatch (Facebook)

Mistake:

“And in this case – it’s a different domain, but I’d say it’s some kind of typo or something like that.”

[P63]
Case 5: Name constraints (Google)

[attendee@devconf ~]$ ./testConnection server_google
Chosen provider: Google
Connecting to accounts.google.com...
Connection success.
Certificate chain saved to server_google.pem.

Certificate validation return value:
permitted subtree violation
(X509_V_ERR_PERMITTED_VIOLATION)
Case 5: Name constraints (Google)

Constraint:

“I understood that there is some chain and a certain point in chain is restricting the hostname to ...” [P39]
Case 5: Name constraints (Google)

Wrong:

“So when I open the certificate, I find out that one of the authorities was listed as false, but the other two were fine.” [P10]
Case 5: Name constraints (Google)

NotKnow:

“I don’t really understand the whole thing.” [P62]
Case 5: Name constraints (Google)

**Attack:**

“*I would probably contact Google and let them know that they have a rogue admin...*” [P26]
Case 5: Name constraints (Google)

CAProblem:

“So while it may have signed that, CA has explicitly said ‘I am not allowed to sign this, you should not trust this.’” [P26]
### Case 5: Name constraints (Google)

<table>
<thead>
<tr>
<th>Score</th>
<th>Issue</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>Constraint*</td>
</tr>
<tr>
<td>19</td>
<td>Wrong</td>
</tr>
<tr>
<td>14</td>
<td>NotKnow</td>
</tr>
<tr>
<td>10</td>
<td>Attack</td>
</tr>
<tr>
<td>10</td>
<td>CAPProblem*</td>
</tr>
<tr>
<td>10</td>
<td>CAConstr*</td>
</tr>
</tbody>
</table>

**CAConstr:**

“The thing is the certificate authority up the chain specifies that only domains with ‘api.google.com’ are valid.” [P18]
Case 5: Name constraints (Google)

**Mistake:**

“It seemed like it was just an innocent misconfiguration of the kind that happens all the time.”

[P19]
Case 5: Name constraints (Google)

25 Constraint*
19 Wrong
14 NotKnow
10 Attack
10 CAProblem*
10 CAConstr*
10 Mistake
10 NolInfo

NolInfo:
“For this one I really try to find some documentation, but there was no documentation on this.”
[P68]
Results II.

Do people in IT Trust flawed certs?
Trust scale (0–6)

6/6: I'm totally satisfied. If it was my bank's website, I would log in without worries.

4/6: Looks OK. I would log in with my library account, but not with my bank account.

2/6: Looks suspicious. I will read the page, but I will not fill in any information.

0/6: Outright untrustworthy. It is not safe to browse or to trust any information there.
Trust comparison by case

- OK: 6/6 “I'm totally satisfied.”
- Expired: 3/6 “Looks OK.”
- Self-signed: 2/6 “Looks suspicious.”
- Hostname mismatch: 1/6
- Name Constraints: 0/6 “Outright untrustworthy.”
Trust in expired certificates (average)

- **1 day**: “Outright untrustworthy.”
- **7 days**: “Looks suspicious.”
- **30 days**: “Looks OK.”
- **365 days**: “I’m totally satisfied.”
Results III.

Do the error messages influence perceptions/trust? (incl. relevant docs)
Idea: Test different designs

A: Original errors
• 44 participants
• OpenSSL 1.1.0g-fips

B: Redesigned errors
• 31 participants
• Our designs
New error messages

OK: All performed check passed.

Expired: The certificate has expired or is not yet valid.

Self signed: The certificate is self-signed and not found in the trust store.

Hostname mismatch: The server hostname does not match the certificate subject name.

Name constraints: The subject name violates constraints set by CA.
[attendee@devconf ~]$ ./testConnection server_google
Chosen provider: Google
Connecting to accounts.google.com...
Connection success.
Certificate chain saved to server_google.pem.

Certificate validation return value:
The subject name violates constraints set by CA.
(X509_ERR_NAME_CONSTRAINTS_VIOLATION,
see https://x509errors.cz)
...leading to x509errors.cz

X509_ERR_HOSTNAME_MISMATCH

The server hostname does not match the certificate subject name.

Explanation

The domain name provided by the server you are connecting to does not match the subject name of the certificate.

Security perspective

Your communication will be encrypted, but you communicate with different (maybe malicious) server than is listed in certificate. However, It can also be caused by malicious attackers pretending to be the server you are connecting to.

Next steps

See the Common Name (CN) or the Subject Alternative Name extension (SAN) in the certificate and compare the value with the domain name of the server. In case of web servers, the error can be caused improper redirect configuration between valid web aliases (e.g. the version of the site without the "www" in domain name).
Caused perception differences

- **OK**
  - More ExtraCheck

- **Self-signed**
  - More Attack

- **Name constraints**
  - More Attack
  - Less codes indicating not understanding (Wrong, NotKnow, NoInfo)
Caused trust differences

- **Self-signed hostname mismatch**
- Name constraints

Decreased trust
Resources used: Just briefly

- Name constraints take longer to comprehend
- People look into certificates (80%)
- Almost everybody googles :-)  
  - 81% with a text code, 66% with own words
- Link offered in the error message clicked often!  
  - 71% of the participants that had it!  
  - Nice opportunity to point users to a useful place
KEEP CALM
there will be
NO MORE RESULTS
Recap: What did we go through?

- Study with 75 DevConf attendees in 2018
- 5 certificate errors (OpenSSL/redesigned)
- Self-signed and name constraints overly trusted?
- Name constraints not much understood
- Expired depends on time elapsed
- Changing errors (& docs) matters
- Links in errors are clicked
What can I do next?

- Submit patches to OpenSSL
  - Error messages
  - Error documentation
- Publish and share results
  - Discussion on cert flaws perception
  - Discussion on name constraints understanding
  - Discussion on links in error messages
What can I do next?

- Map errors of different libraries
  - Do similar errors mean the same things?
  - Compare/share documentation
- Unify errors and documentation
  - Parallel with web world:
    2017-10: Mozilla, Microsoft, Google, W3C, Samsung create cross-browser documentation on MDN
What can YOU do next?

• Read error messages in your product.
  – Do the users/developers understand them?
  – Ask them! (Or make a study.)

• Like the ideas presented here?
  – Spread the word. The paper will be available soon.
    (Sign up for a notification if you want.)
  – Share feedback in person or by email.
Usable security may still be unusual...

We use TLS certificate validation as a real-world example to spark conversation on usable security and developer experience. This research is a part of the academic cooperation of Red Hat and Masaryk University.
Usable security may still be unusual...

We use TLS certificate validation as a real-world example to spark conversation on usable security and developer experience. This research is a part of the academic cooperation of Red Hat and Masaryk University.

Unusual word pair

It seems that the noun security might combine better with an adjective other than usable. Consider rewriting this word pair or choosing a synonym for usable.

good
proper
May your software always be usable!
(and secure!)

Interested in the research?
My other research bits at crocs.fi.muni.cz/people/mukrop

Martin Ukrop, mukrop@mail.muni.cz
Masaryk University, CRoCS
Ph.D. research cooperation with Red Hat Czech