Improving SNI-based HTTPS Security Monitoring

Wazen Shbair, Thibault Cholez, Jerome Francois, Isabelle Chrisment

LORIA, UMR 7503, University of Lorraine, INRIA, France

Thibault Cholez
thibault.cholez@loria.fr

Second IEEE International Workshop on Security Testing and Monitoring
Workshop of IEEE ICDCS 2016,

Nara, Japan - 27 June 2016

Outline

HTTPS Traffic Dilemma

HTTPS Traffic

- Encryption is the commonly used solution to guarantee privacy and security.
- Based on an ARCEP survey [1]: HTTPS accounted for 5% of Internet traffic in 2012, 50% in 2015.

The Dilemma

- Content providers need to secure content over the Web.
- Network administrators may need to monitor/filter access to some HTTPS websites.

Research question

How can we properly monitor HTTPS traffic?

HTTPS Traffic Monitoring

Legacy solutions don't work

- Port Based, DNS and IP address are not reliable [2].
- Deep Packet Inspection (DPI): Encrypted traffic challenge [3].

Practical solutions have drawbacks

- Certificate Filtering: single certificate for multiple domains [4].
- HTTPS Proxy: It's hardly acceptable to trust third-party to screen sensitive information.

A recent/alternate method to Monitor HTTPS:

Using Server Name Indication (SNI) extension for HTTPS traffic identification and filtering.

Overview of SNI

What is SNI ?

- SNI is an extension inside Client Hello Message.
- SNI originally proposed to support virtual hosting for websites using HTTPS [5].
- SNI helps the server for mapping between the requested domain from the client/browser and the corresponding SSL/TLS server certificate [6].

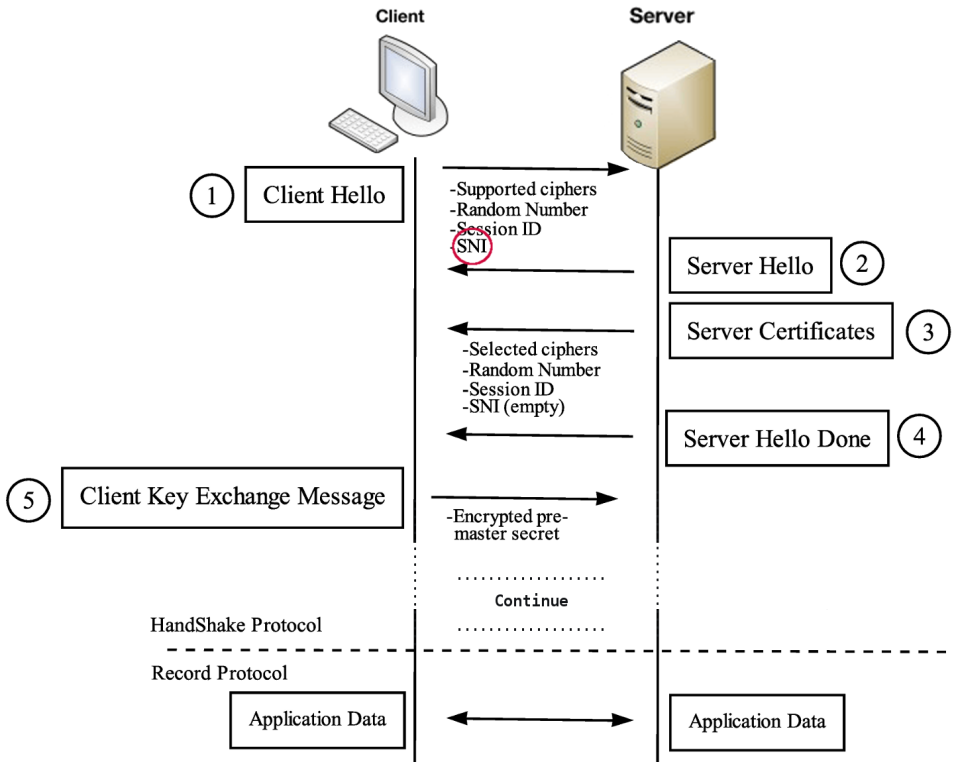


Figure : TLS Handshake

Overview of SNI

Why SNI ?

- SNI is a simply extracted value from the extensions list.
- All recent Web browsers and servers OS support SNI [5].
- Users' privacy is untouched.
- Implemented in many firewalls (Sphirewall, Untangle NG, IPFire, etc.).

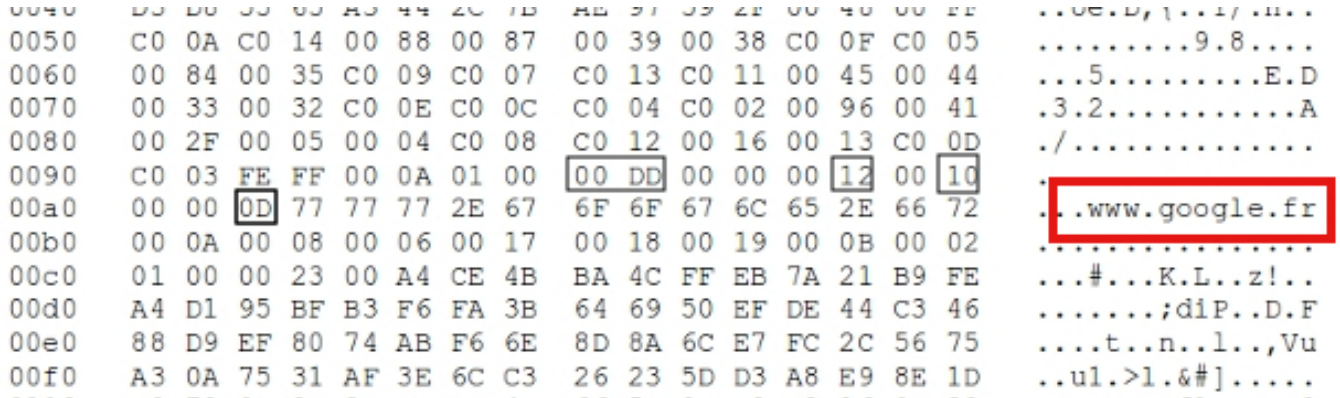


Figure : Server-name in SNI Extension

SNI-Based HTTPS Filtering

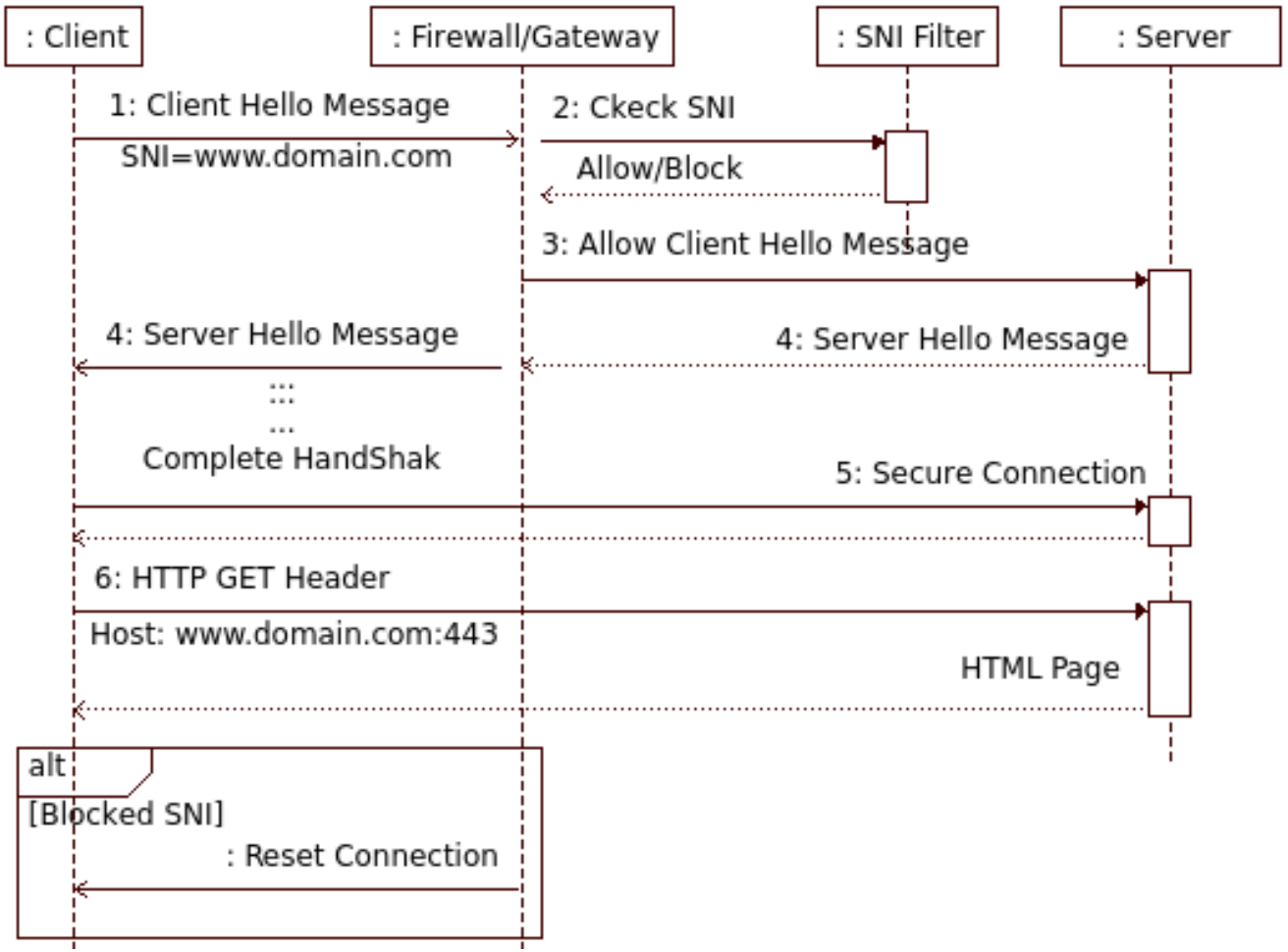


Figure : SNI Filtering Sequence Diagram

SNI-Based HTTPS Filtering

What about the reliability of SNI-Filtering ?

- Is it possible to bypass this type of filtering?
- The usage of SNI for identifying HTTPS traffic needs to be assessed.

Strategies Exploiting SNI-Filtering Weakness

1. Backward Compatibility

Based on RFC6066, states that TLS clients that support the SNI extension can still talk to servers that do not support it, and vice versa [6].

2. Shared Server Certificate

The alternative name field in the certificate standard X.509 makes it possible to hold a set of domain names using the same certificate [4].

Bypassing Firewall Systems

These weaknesses can be used for circumventing firewalls relying on SNI to monitor and filter HTTPS traffic.

1. Backward Compatibility

Backward compatibility can be used to cheat firewalls as follow:

1. Remove SNI extension form the client-hello.
2. Insert an alternative/fake "server-name" in the SNI.

Example: assume Facebook is blocked

- Create TLS Java Socket :
  `TARGET_HTTPS_SERVER=facebook.com`
  `TARGET_HTTPS_PORT=443`
- Create SNI Object with `server_name=f@ceb00k.com`
- Send HTTP host header with real address of the blocked website:
  `GET/HTTP/1.1/r/n`
  `HOST:facebook.com:443`

2. Shared Server Certificate

This can be used to get access to a banned website by sending the SNI for non-banned websites sharing the same server certificate

Example: assume Youtube is blocked

- Create TLS Java Socket :
  `TARGET_HTTPS_SERVER=maps.google.com`
  `TARGET_HTTPS_PORT=443`
- Create SNI Object with `server_name=maps.google.com`
- Send HTTP host header with real address of the blocked website:
  `GET/HTTP/1.1/r/n`
  `HOST:youtube.com:443`

Implementation of a Web Browser Plug-in

Escape

- Escape[1] is an add-on for the Firefox web browser.
- It can get the control over the TLS handshake and hack the SNI value.
- The motivation is the strong relation between our work and web browsing.

How does it work?

Like a **LOCAL** proxy that intercepts TLS connections and creates its own with other parameters.

[1] http://madynes.loria.fr/Research/Software

Implementation of a Web Browser Plug-in

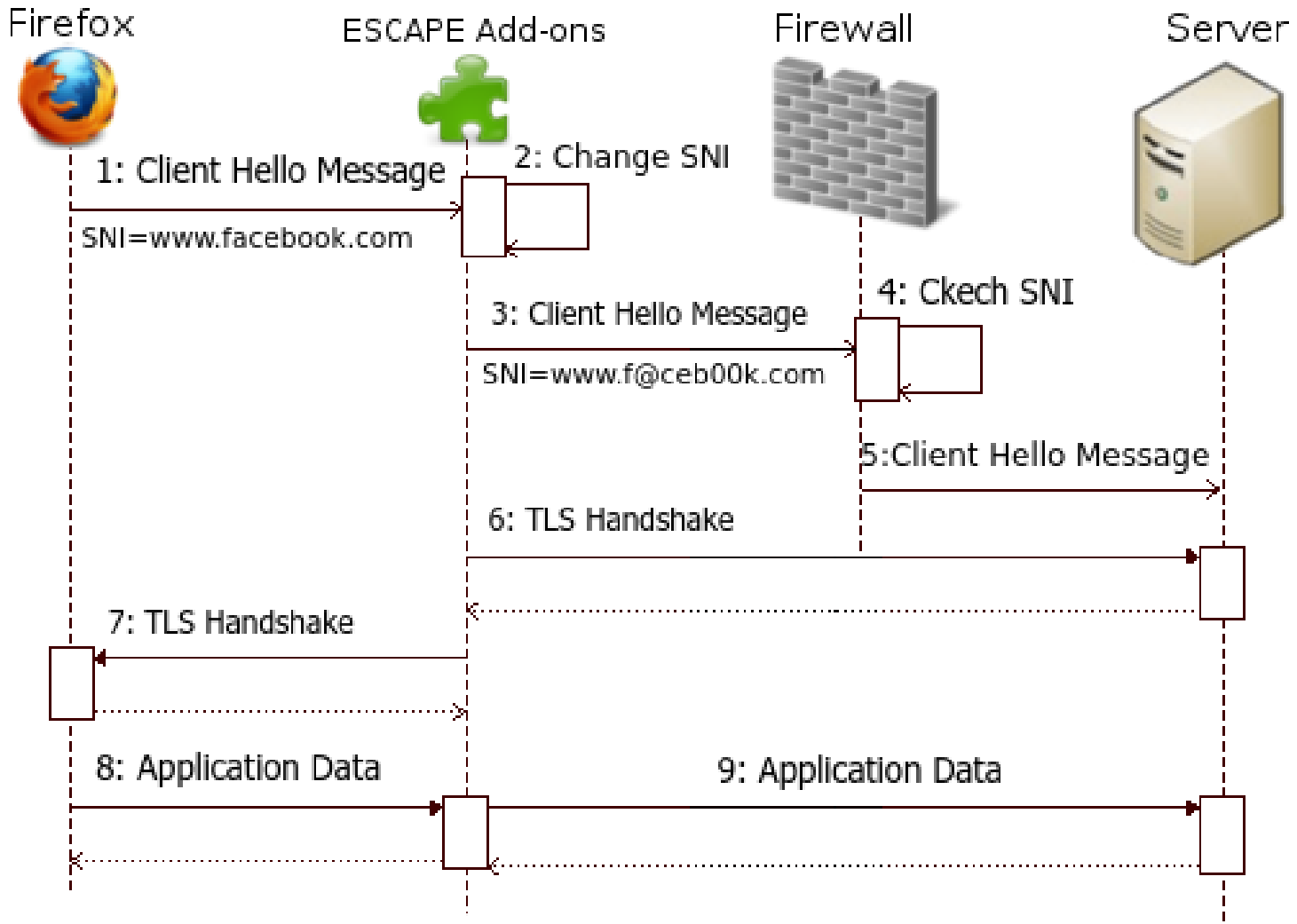


Figure : Escape plug-in interaction


Esqape
BYPASSING HTTPS FILTERING

Evaluation of the bypassing strategy

- We tested 3 firewalls filtering on SNI (Sphirewall, IPFire, Untangle NG Firewall)
- We investigated the Top 500 HTTPS websites (Alexa) + 14 others sensible HTTPS websites.
- Two distinct criteria are considered for each HTTPS server:
  1. the support of SNI
  2. the behavior when a fake SNI is received

Results

- None of the tested firewalls were able to detect fake SNI.
- 44% of servers do not support SNI (as specified by RFC).
- 8% of websites can't establish the TLS connection with a fake SNI *"HTTP hostname and TLS SNI hostname mismatch"*.
- 92% of websites are successfully accessed with a fake SNI.

Remediation

How to detect a Forged-SNI ?

- We can not totally rely on the server side to detect forged-SNI (backward compatibility).
- How to verify the content and the veracity of the SNI extension?

DNS to the rescue

Assessing SNI with DNS service

- According to the RFC[6], SNI must only contain DNS-resolvable hostnames
- Use a trusted DNS server to check the correspon -dence between the destination server IP addr and the DNS response.
- Fake SNI (random string or wrong server) will create inconsistencies.

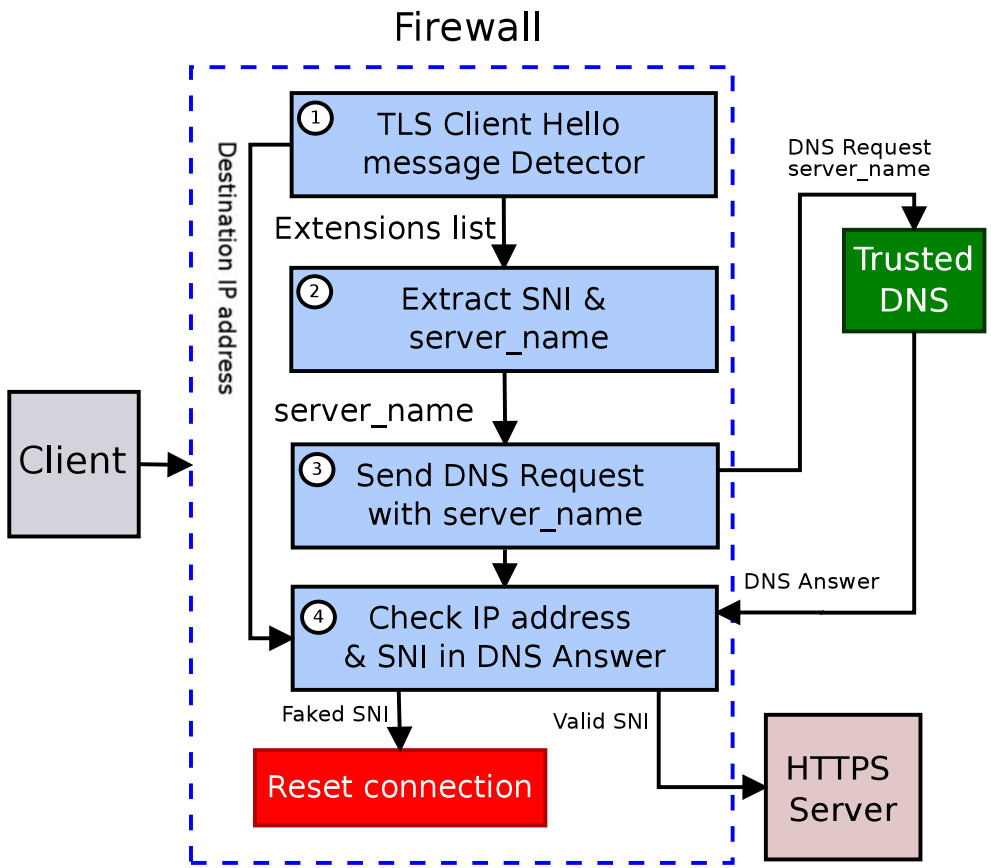


Figure : Client SNI verification using DNS

DNS to the rescue

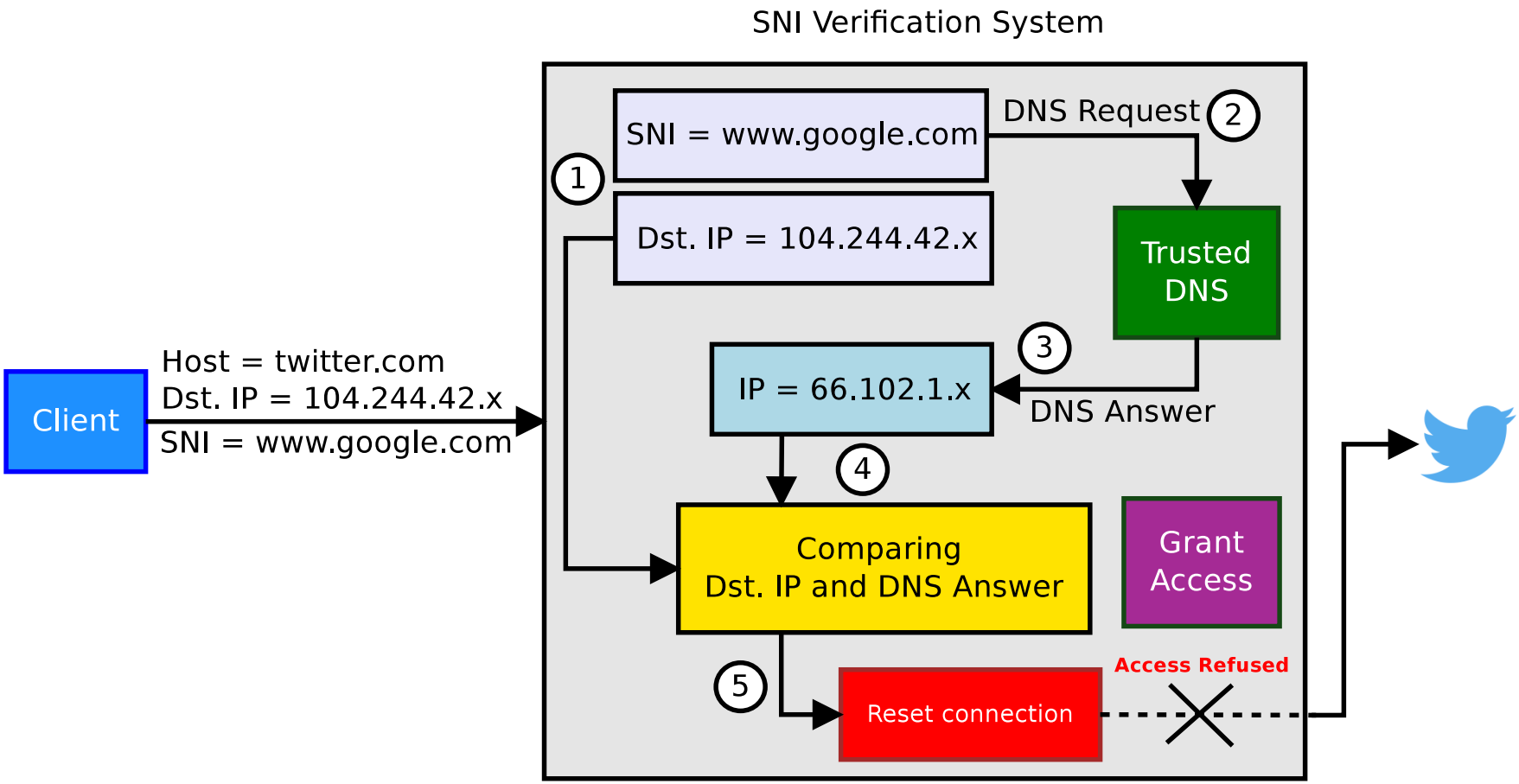


Figure : SNI verification using DNS

Improvement Evaluation: False positive rate

Methodology

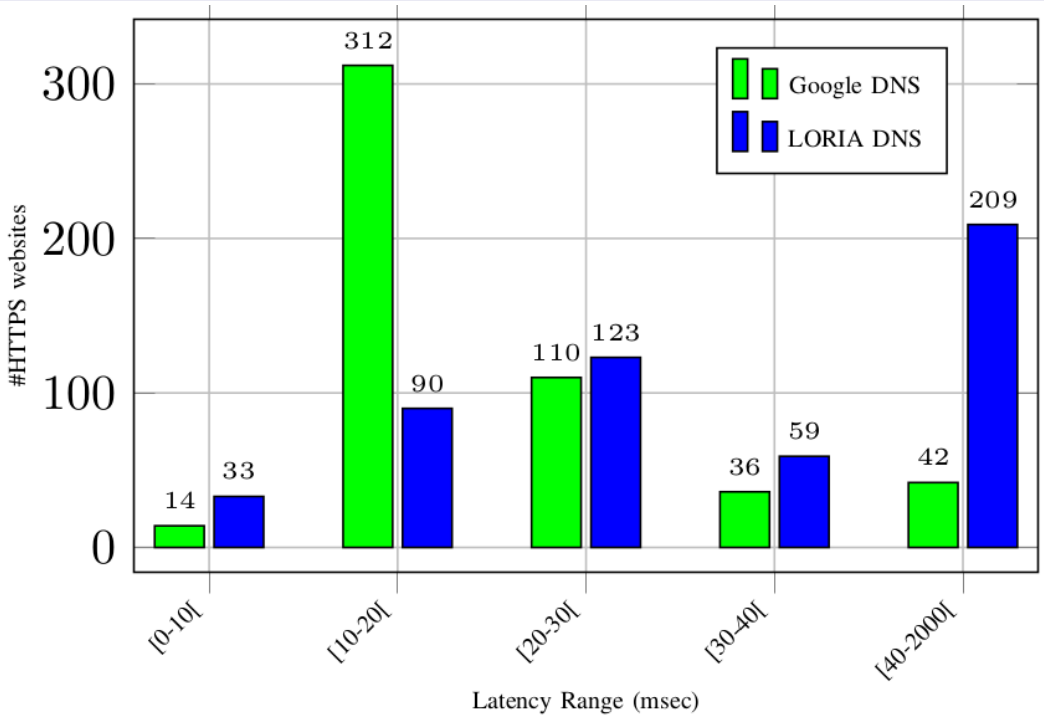
- DNS responses can be inconsistent in time [7] (load balancing, etc.) → Need to assess the classification of legit HTTPS connections (no fake SNI).
- We use 2986 HTTPS connections related to the previous 514 HTTPS web sites (including embedded content, etc.).
- Results: all websites are accessible, FP rate depends on the matching strategy.

Detection Strategy	# Connections	True Negatives	False Positives
Exact match	2501	83.75%	16.25%
First 24-bits	2771	92.79%	7.21%
First 16-bits	2935	98.29%	1.71%

Improvement Evaluation: overhead

Additional delay induced by DNS requests

- Mainly depends on the DNS server. With Google DNS, average added delay is less than 15 ms.
- For comparison:avg. HTTPS server response time ~483ms [8].



Conclusion and Future Works

Conclusion

- SNI-based HTTPS filtering has two weaknesses: backward compatibility and multiple services using a single certificate.
- 92% of HTTPS websites are accessible with a fake SNI (Escape tool).
- A trusted DNS can help to assess SNI values.

Future Work

- Identify HTTPS services in a more robust way, based on traffic pattern.
- Promising results [9]. Next goal: real time identification.

References

[1] "Report 2015-0832 from the French regulatory authority for telecommunications (ARCEP)." [In French], Accessed: 19/04/2015.

[2] S. Valenti, D. Rossi, A. Dainotti, A. Pescap, A. Finamore, and M. Mellia, "Reviewing traffic classification," in *Data Traffic Monitoring and Analysis* (E. Biersack, C. Callegari, and M. Matijasevic, eds.), vol. 7754 of *Lecture Notes in Computer Science*, pp. 123–147, Springer Berlin Heidelberg, 2013.

[3] Y. Xue, D. Wang, and L. Zhang, "Traffic classification: Issues and challenges," in *2013 International Conference on Computing, Networking and Communications (ICNC)*, pp. 545–549, 2013.

[4] J. Clark and P. C. van Oorschot, "SoK: SSL and HTTPS: revisiting past challenges and evaluating certificate trust model enhancements," in *IEEE Symposium on Security and Privacy*, pp. 511–525, IEEE, 2013.

[5] L. Vlker, M. Noe, O. P. Waldhorst, C. Werle, and C. Sorge, "Can internet users protect themselves? challenges and techniques of automated protection of HTTP communication," in *Computer Communications*, vol. 34, pp. 457–467, 2011.

[6] D. Eastlake, "RFC 6066 - the transport layer security (TLS) extensions: Extension definitions," 2011.

[7] T. Mori, T. Inoue, A. Shimoda, K. Sato, K. Ishibashi, and S. Goto, "SFMap: Inferring services over encrypted web flows using dynamical domain name graphs," in *Traffic Monitoring and Analysis: 7th International Workshop, TMA 2015 Proceedings*, pp. 126–139, Springer, 2015.

[8] M. Prandini, M. Ramilli, W. Cerroni, and F. Callegati, "Splitting the HTTPS stream to attack secure web connections," *IEEE Security and Privacy*, vol. 8, no. 6, pp. 80–84, 2010.

[9] W. Shbair, T. Cholez, J. Francois, and I. Chrisment, "A multi-level framework to identify https services," in *IEEE/IFIP Network Operations and Management Symposium*, 2016.