Automating Compliance for Improving TLS Security Postures: An Assessment of Public Administration Endpoints

Riccardo Germenia[1,2][a], Salvatore Manfredi[2][b], Matteo Rizzi[1,2][c], Giada Sciarretta[2][d], Alessandro Tomasi[2][e] and Silvio Ranise[2,3][f]

[1] *Department of Information Engineering and Computer Science, University of Trento, Via Sommarive 9, Trento, Italy*
[2] *Center for Cybersecurity, Fondazione Bruno Kessler, Via Sommarive 18, Trento, Italy*
[3] *Department of Mathematics, University of Trento, Via Sommarive 14, Trento, Italy*

Keywords: Compliance Analysis, National Guidelines, Auditable Dataset, TLS Deployments.

Abstract: System administrators tasked with configuring TLS servers must make numerous decisions - e.g., selecting the appropriate ciphers, signature algorithms, and TLS extensions - and it may not be obvious, even to security experts, which decisions may expose them to attacks. To address this issue, raise awareness, and establish a security threshold, numerous cybersecurity agencies around the world issue technical guidelines for the use and configuration of TLS. In this paper we carry out an assessment of the TLS security posture of European and US based endpoints in relation to their respective national cybersecurity guidelines. Our results show that a surprisingly high amount of the analyzed websites have a low compliance level when compared to their respective national guideline. We attempt to identify potential causes by presenting a series of observations that may underlie the lack of compliance. The analysis is conducted by employing a TLS analyzer we developed to automate the compliance analysis and the application of the suggested changes, assisting system administrators during this important yet complex task. Our tool and the dataset containing the machine-readable requirements for automating conformity assessment are publicly available, thus making the process auditable and the assets extensible.

1 INTRODUCTION

Transport Layer Security (TLS) provides confidentiality and integrity between communicating entities. It is mainly used to secure online traffic by preventing eavesdropping and tampering. Administrators in charge of configuring TLS servers have many choices to make - e.g., choose the right ciphers, signature algorithms and TLS extensions - and it may not be obvious even to experts which choices may lead to exposing vulnerabilities. As an example, the confidentiality of more than 170,000 websites (Shodan Search Engine, 2024) can still be compromised using Heartbleed, an attack discovered in 2014 that exploits an insecure implementation of the Heartbeat TLS extension (Synopsys, Inc, 2014).

Based on RFCs and IANA public registries for standards and parameters (IANA, 2005b; IANA, 2005a), authorities such as the European Commission, the White House Office of Management and Budget, and cybersecurity agencies such as US NIST and Italian AgID issue guidelines to define how a deployment should be configured to avoid insecure configurations. These guidelines are documents that establish a level of security by listing requirements. Theoretically, each guideline can specify a requirement for every configurable element of a webserver - e.g., for the NIST guideline, TLS version 1.2 `MUST` be supported - by taking into account published standards, deprecated features, and known attacks. The following is an example of a human-readable requirement extracted from *NIST SP 800-52 Rev. 2* (National Institute of Standards and Technology (NIST), 2019):

[a] https://orcid.org/0009-0009-9283-8030
[b] https://orcid.org/0000-0001-9645-6034
[c] https://orcid.org/0000-0002-5288-3031
[d] https://orcid.org/0000-0001-7567-4526
[e] https://orcid.org/0000-0002-3518-9400
[f] https://orcid.org/0000-0001-7269-9285

Germenia, R., Manfredi, S., Rizzi, M., Sciarretta, G., Tomasi, A. and Ranise, S.
Automating Compliance for Improving TLS Security Postures: An Assessment of Public Administration Endpoints.
DOI: 10.5220/0012764700003767
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 21st International Conference on Security and Cryptography (SECRYPT 2024)*, pages 450-458
ISBN: 978-989-758-709-2; ISSN: 2184-7711
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

Table 1: TLS protocol compliance across guidelines (dataset excerpt).

	NIST (user-facing)	BSI (federal)	BSI (user-facing)	ANSSI	AgID
TLS 1.0	Not recommended	Must not	Not recommended	Must not	Not recommended
TLS 1.1	Not recommended	Must not	Not recommended	Must not	Not recommended
TLS 1.2	Not recommended	Must	Recommended	Optional	Must
TLS 1.3	Must	Recommended	Recommended	Recommended	Must

Servers that support citizen or business-facing applications [...] shall be configured to negotiate TLS 1.2 and should be configured to negotiate TLS 1.3. The use of TLS versions 1.1 and 1.0 is generally discouraged, but these versions may be configured when necessary to enable interaction with citizens and businesses. [...] These servers shall not allow the use of SSL 2.0 or SSL 3.0.

Despite raising the security bar and being able to prevent known attacks, allowing system administrators to improve the security posture of their services, compliance with these recommendations is not required but strongly suggested. Non-compliance with these guidelines may result in a reduction of the minimum security level recommended by technical agencies, fines (European Parliament, 2016; Office of Management and Budget, 2017), or even exclusion from international payment systems (i.e., PCI-DSS (PCI, 2018)).

While the existence of technical guidelines can help raise awareness and set a minimum security level, their use is nontrivial as a system administrator must examine each element covered by the set of requirements, understand the language and the way the requirement is written and (if clear) verify that its value matches the expected value. The situation worsens when a service needs to be provided across multiple countries. Since each national authority independently sets different requirements for each configurable element, a given configuration may render a system fully-compliant with one state guideline while falling below a desired threshold in another.

System administrators could likely benefit from automated assistance in their decision-making, during the process of configuring a service to be compliant against a given guideline. To accomplish this, we collected a set of compliance requirements - from different national cybersecurity agencies - to build a machine-readable knowledge base which can be used to streamline the process of configuring a compliant webserver. Moreover, we have developed a *Compliance Analyzer Module* that allows to automate the verification process. We validated its functioning by assessing the security posture of European and US based public administration endpoints.

Our contributions can be summarized as follows:

- publication of an auditable dataset of technical requirements extracted from national security guidelines issued by NIST (United States), BSI (Germany), ANSSI (France) and AgID (Italy), which references Mozilla;
- implementation of a compliance module for TLSAssistant[1] that employs the extracted technical guidelines to automate the compliance analysis and provide actionable hints on how to attain the desired compliance; and
- an assessment of the TLS compliance level of European and US based webservers respect to their national cybersecurity guidelines. We also provide observations to investigate the reason for any observed lack of compliance.

Plan of the Paper. Section 2 provides a brief description of the methodology we designed to compare a webserver posture against single and multiple guidelines, illustrating a set of use cases and how to handle requirements conflicts. Section 3 outlines the methodology employed to determine the choice of the endpoints, as well as providing a description of a classification system utilized for organizing the resulting data. The experimental results, accompanied by a series of observations that attempt to provide insight into the potential reason for the obtained data, are detailed in Section 4. Section 5 presents related work and Section 6 concludes the paper and highlights future work.

2 GUIDELINES AND COMPLIANCE ON TLS

With the aim of helping system administrators make their services compliant with agencies' guidelines, we published an auditable dataset (*TLS_Compl_Dset*) containing requirements that must be managed to ensure compliance. These requirements were gathered,

[1]TLSAssistant is an open-source modular framework capable of identifying a wide range of TLS vulnerabilities. Its actionable report can assist the user in correctly and easily fixing their configurations. Its source code is available on https://github.com/stfbk/tlsassistant

Table 2: Compare-to-one methodology.

Requirement level	Compare-to-one	
	Element detected	Element missing
MUST	Do nothing	ERROR: enable
MUST NOT	ERROR: disable	Do nothing
RECOMMENDED	Do nothing	ALERT: enable
NOT RECOMMENDED	ALERT: disable	Do nothing
OPTIONAL	Do nothing	Do nothing
Not explicitly mentioned	Guideline-dependant	Do nothing

translated, standardized, and organized taking information from five guidelines[2] published by NIST (National Institute of Standards and Technology (NIST), 2019), BSI (Bundesamt für Sicherheit in der Informationstechnik (BSI), 2024), ANSSI (ANSSI, 2020), AgID (Agenzia per l'Italia Digitale, 2020), and Mozilla (Mozilla Foundation, 2023).[3] The dataset provides information on the source document, requirement level[4], any additional indications specified by the agencies, and a set of notes explaining the reasoning behind each type of extraction, for each available category (e.g., cipher suites and TLS extensions).

TLS_Compl_Dset is made available on (tls, 2024) to facilitate verification of its contents, enabling independent and transparent validation, as well as receiving suggestions for enhancing the data.

When taking into consideration a *single* guideline (compare-to-one), the act of evaluating if a deployment is compliant with a given set of requirements can be performed by scrolling through each requirement and acting accordingly to the requirement level. Table 2 briefly depicts the proposed methodology. The first column indicates the requirement level an agency assigns to a given element - e.g., the presence of a specific extension, the availability of a given protocol or cipher and the next two columns describe how to behave if a given element is detected or missing, respectively, in the target configuration.

As an example, let us focus on protocol compliance. If a system administrator wants to make a deployment compliant with BSI (federal) (see the third column in Table 1), and supposing that the webserver currently offers both TLS 1.1 and 1.3, our methodology would guide the system administrator towards:

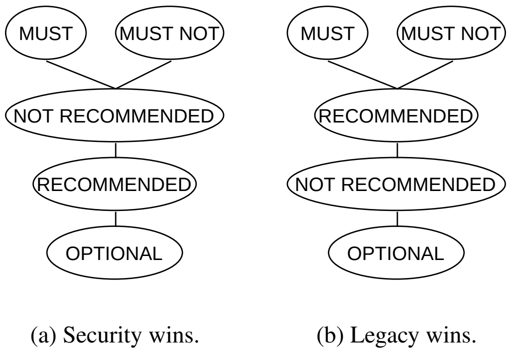


Figure 1: Defined partial orders.

- disabling TLS 1.1 as it `MUST NOT` be available in any circumstance;
- enabling TLS 1.2 as it `MUST` be available;
- keep TLS 1.3 enabled as it is `RECOMMENDED`.

Comparing *multiple* guidelines (compare-to-many) is not obvious as RFC 2119 (Bradner, 1997) does not provide any relation among the requirement levels, thus making it impossible to directly compare two of them without defining a consistent methodology.

Indiscriminately considering any `RECOMMENDED` requirement stronger than a `NOT RECOMMENDED` one (i.e. `RECOMMENDED` $\succ$ `NOT RECOMMENDED`) may lead to overly strict configurations, whereas doing the opposite (i.e. `NOT RECOMMENDED` $\succ$ `RECOMMENDED`) would result in configurations containing unsecure or deprecated features. For this reason, we define two posets[5] to allow a configuration to reflect custom preferences, leaving the system administrator the choice to use one of the following comparison approaches:

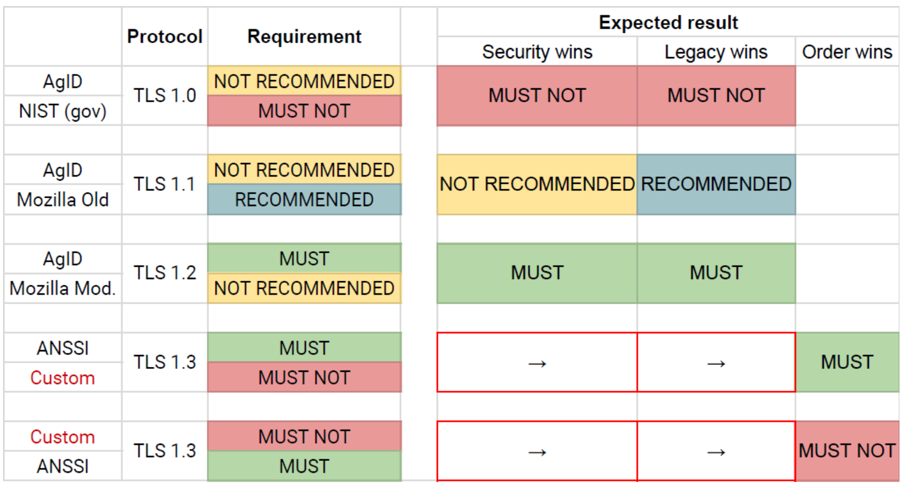
Security Wins. the user opts for a more restrictive configuration with the intention of maximizing the security, potentially reducing the service availability to legacy systems. This approach favors `NOT RECOMMENDED` requirements over the `RECOMMENDED` ones;

[2]The guidelines have been chosen for their heterogeneity as they employ different languages, data structuring and keywords (Manfredi, 2023). Other national security guidelines exist (e.g., from the Netherlands National Communications Security Agency - NLNCSA) and may be covered in future work.

[3]Although not being published by a national cybersecurity agency, the Mozilla guidelines were chosen because they are explicitly included in the AgID guidelines.

[4]Standardized using RFC 2119 (Bradner, 1997) set of keywords.

[5]A poset $P = (S, \prec)$ consists of a set S and a binary relation $\prec$ that orders certain pairs of elements - i.e., a partial order.



	Protocol	Requirement	Expected result		
			Security wins	Legacy wins	Order wins
AgID	TLS 1.0	NOT RECOMMENDED	MUST NOT	MUST NOT	
NIST (gov)		MUST NOT			
AgID	TLS 1.1	NOT RECOMMENDED	NOT RECOMMENDED	RECOMMENDED	
Mozilla Old		RECOMMENDED			
AgID	TLS 1.2	MUST	MUST	MUST	
Mozilla Mod.		NOT RECOMMENDED			
ANSSI	TLS 1.3	MUST	→	→	MUST
Custom		MUST NOT			
Custom	TLS 1.3	MUST NOT	→	→	MUST NOT
ANSSI		MUST			

Figure 2: Comparison approaches.

Legacy Wins. the user chooses a configuration that is more flexible in order to allow access to the service for the greatest number of systems, even those that are not modern. This approach favors `RECOMMENDED` requirements over the `NOT RECOMMENDED` ones.

The orders can also be represented using Hasse diagrams (see Figure 1), graphical representations in which every comparable element of the set is connected to another using a segment that indicates a relation. Hasse diagrams have an implied upward orientation: if $x \prec y$, x appears below y in the drawing. For example, in the Hasse diagram representing the *Security wins* approach, `OPTIONAL` appears lower than `RECOMMENDED` because `OPTIONAL` $\prec$ `RECOMMENDED`.

In both orders, `OPTIONAL` is the least element as it precedes all other elements. This means that, when compared to any element with a higher position, the `OPTIONAL` recommendation is ignored. Meanwhile, `MUST` and `MUST NOT` are the maximal elements, as they succeed all other elements.

`MUST` and `MUST NOT` are not part of the defined ordering as they are both absolute requirements, thus they cannot be compared in an objective way, and consequently lack a segment that connects them in Figure 1. With the guidelines taken into consideration here, no requirement evaluation can result in a `MUST` vs. `MUST NOT` comparison. However, to make our methodology future-proof and help to manage different scenarios, we introduce the concept of a *custom* guideline: this may be used to compare an organization's security policy, or to narrow an existing guideline by taking into account additional threat assumptions. Then, to provide a consistent way to manage multiple guidelines taking into account any possible comparison, we propose the following approach:

Order Wins. the user explicitly chooses which authority should take precedence when comparing their set of recommendations. By doing so, the first authority's decisions on `MUST` and `MUST NOT` requirements directly reflects on the final output, ignoring the second authority choice.

Its usage with different requirements is shown in Figure 2. Taking as example TLS 1.3, its `MUST` requirement level would lose against a `MUST NOT` if placed second but will win if placed first.

In case of an element not mentioned in the guideline, we extend the approach illustrated in Table 2 and consider any missing requirement level as an `OPTIONAL`. By doing so, the element not covered by the guideline cannot overcome the defined one, but it is still taken into consideration for specific edge cases: for example when a given element appears in a configuration (*compare-to-many* scenario) but is not covered by both guidelines.

3 EXPERIMENTAL EVALUATION

The following sections describe the TLS compliance evaluation of European and U.S. public administrations. The objective is to capture a snapshot of the current adherence to various agencies' indications and attempt to extract some observations based on the resulting data. We detail: *(i)* the process of homogeneously selecting target websites, belonging to different states and *(ii)* a description of the classification system utilized for organizing the resulting data.

Table 3: Fulfilled requirements across public administration websites.

Requirements category	DE - 16 hosts, BSI			FR - 11 hosts, ANSSI			IT - 14 hosts, AgID			US - 15 hosts, NIST		
	NC	NR	C	NC	NR	C	NC	NR	C	NC	NR	C
Protocol	0	1	15	2	0	9	0	6	8	3	12	0
CipherSuite	0	0	16	0	9	2	0	0	14	13(d)	1	1
CipherSuitesTLS1.3	0	0	16	0	0	11	0	0	14	12	0	3
Extension	0	0	16	0	11	0	1	0	13	15	0	0
Groups	0	0	16	2	0	9	0	9	5	0	0	15
Signature	0	0	16	11(a)	0	0	N/A	N/A	N/A	15(a)	0	0
Hash	0	0	16	10(b)	0	1	N/A	N/A	N/A	15(b)	0	0
CertificateSignature	0	0	16	0	0	11	-	-	-	0	0	15
KeyLengths	14(c)	0	2	0	0	11	0	0	14	0	0	15
Certificate	1	0	15	N/A	N/A	N/A	N/A	N/A	N/A	0	0	15
CertificateExtensions	4	0	12	2	0	9	N/A	N/A	N/A	15	0	0
Misc	0	0	16	N/A	N/A	N/A	0	4	10	0	0	15

3.1 Target Selection

To find targets for our evaluation, we focused on a common feature that all countries mentioned have: government departments or ministries. This allows us to focus on the highest levels of the government hierarchy, which we may infer are the most crucial locations to protect and the most vulnerable endpoints (due to their high visibility). We assumed that a ministry or government department website would be more likely to adhere with compliance regulations in comparison to a small city.

France. For each entry in the official government website (Government of France, 2024) of each ministry head, we obtained the corresponding ministry and obtained a total of eleven official websites.

Germany. We found sixteen official websites for the German Federation by accessing the “List of Federal Ministries” page (Domestic Protocol Office of the Federal Government, 2023) on the website of the Domestic Protocol Office for the *Bundesrepublik Deutschland*.

Italy. A search for the term “Ministero” in the “Enti” segment of the Open Data AGID dataset (Agenzia per l’Italia Digitale, 2024) returned fourteen results.

US. A lookup for “Department” in the *usa.gov* directory of federal agencies and departments (USAGov, 2024) yielded fifteen results.

3.2 Websites Analysis and Data Elaboration

To carry out a bulk analysis and capture a snapshot of the compliance status at the time of execution, we run the *Compliance Analyzer Module* (which we have built in TLSAssistant) on the list of 56 chose websites associated to the corresponding guideline - e.g., Italian websites compared to AgID guidelines, German to BSI ones - using the `compare-to-one` submodule.

We classified the resulting data into three distinct compliance categories, inspired by the notion of compliance and the posets introduced in Section 2. For each of the twelve configurable element categories (e.g., protocols and signature algorithms), a website would be labeled as:

Not compliant (NC). if at least one of the absolute requirements (`MUST` and `MUST NOT`) is not met. This choice is driven by the requirement listed as absolute (i.e., cannot be affected by technical limitations or user choices), and a direct neglection can be considered an issue.

Not recommended (NR). if all the absolute requirements are met but any of the `NOT RECOMMENDED` ones is not. This choice is driven by the idea that the system administrator may have decided to relax a requirement for technical reasons (e.g., legacy hardware or software).

Compliant (C). if all absolute requirements (`MUST` and `MUST NOT`) and all `NOT RECOMMENDED` requirements are satisfied.

We show the full set of categorized data from observations in Table 3.

4 EXPERIMENTAL FINDINGS AND INTERPRETATION

The availability of security guidelines can pave the way towards achieving a unified security posture. However, the analyzed guidelines do not specify the precise manner in which accomplish particular objectives; this aspect, which appears frequently in the literature (Manfredi et al., 2022; Acar et al., 2016; Gorski et al., 2018; Krombholz et al., 2017; Tiefenau et al., 2020; Li et al., 2019), should not be overlooked as it places a significant burden on the shoulders of system administrators.

The resulting data[6] (see in Table 3) show that only a subset of these recommendations are effectively adhered to, while others appear to be more difficult to apply for reasons that can only be deduced. Most of the verified websites (75%) fall into the *Not compliant*, leaving few sites in the *Not recommended* category (∼21%) and only two websites (∼4%) in the *Compliant* one. This is likely to happen because the absolute requirements demand more impactful changes (especially in the `MUST NOT` case) and thus, after managing them, the handling of `NOT RECOMMENDED` requirements is relatively easy to address. By considering the outlier data, we can also provide the following observations:

Missing Signature Algorithms Compliance (ANSSI and NIST). The websites denoted by the symbol *(a)* (see Table 3) have been determined to be "Not compliant" due to the discovery, through careful review, that they offer *rsa_pkcs_* as signature algorithm. According to ANSSI TLS guideline v1.2 (ANSSI, 2020), the use of these algorithms is labeled as `MUST NOT` starting from 2020. While, NIST guidelines (National Institute of Standards and Technology (NIST), 2019) classified their use as `NOT RECOMMENDED` prior to 31 Dec 2023 and `MUST NOT` as of January 1, 2024.

A possible cause for this non-compliance with the guidelines is the inclusion of those signature algorithms in the list of those that are enabled by default in OpenSSL. Subsequently, a system administrator running a deployment with default values may inadvertently provide them, thereby rendering the system non-compliant. As the available hash algorithms depend on the choice of signature algorithms selected, the two categories are intertwined (as represented by the symbol *(b)*).

Missing Keylenghts Compliance (BSI). The websites denoted by the symbol *(c)* are deemed "Not compliant" due to the use of DH2048 and RSA2048 keylength mechanisms. These two mechanisms were change from `RECOMMENDED` to `MUST NOT` in January 1, 2023 and 2024 respectively.

We surmise this issue may be due at least in part to the content of a guideline specifying a switch-over from a certain date, but the published guideline itself not being published in a new version.

Missing Compliance Categories (AgID). AgID guidelines are less specific than their counterparts, so websites run by the Italian public administration do not face significant issues. Furthermore, AgID guidelines do not cover certificate management. If compared to the NIST guidelines, Italian websites would be equally problematic since they use the same default values that render "Not compliant" the websites indicated in Table 3 with *(a)* and *(b)*.

Missing Cipher Suite Compliance (NIST). We determined that fourteen of the fifteen websites based in the United States are deemed "Not compliant" (denoted by the symbol *(d)* in Table 3) because they offer the CHACHA20_POLY1305 cipher suite. This is a peculiar issue because the NIST guidelines (National Institute of Standards and Technology (NIST), 2019) instructs that all cipher suites not explicitly listed in the document MUST NOT be considered; therefore, the utilization of CHACHA20_POLY1305 is prohibited by default.

Similar to items *(a)* and *(b)*, the availability of these cipher suites may be attributed to the use of a default OpenSSL configuration.

5 RELATED WORK

There exist multiple TLS analyzers that also perform a compliance evaluation. However, their functioning is limited to a single guideline and the reports often lack clarity as they do not specify which changes are mandatory (e.g., `MUST`) and which offer some degree of freedom (e.g., `RECOMMENDED`). In addition, none of the presented tools offer actionable hints on how to attain the desired compliance. A brief comparison of covered guidelines and features is shown in Table 4.

- **sslyze** (Diquet, 2023)**:** is an open source downloadable tool that can be used to check if a server uses strong encryption settings and if it is vulnerable to a subset of known TLS attacks. Since version 5.0.0, published in November 2021, sslyze is able to check a server's configuration against Mozilla recommended configurations (Mozilla Foundation, 2023). Its output shows a list of bullets containing the simple list of unmet requirements. It only covers one of the four identified use cases (i.e. compare-to-one) and only evaluates compliance against the Mozilla guideline.
- **TLS Profiler** (Fett, 2023)**:** is an analysis tool able to compare a server's configuration against Mozilla's profiles. While being based on sslyze, TLS Profiler implementation predates the former as its development started on September 2019. The tool, which can both be downloaded or used

[6]The replication package - that can be employed to repeat the analysis - can be found in (Germenia et al., 2024).

Table 4: Comparison between TLS compliance analyzers.

Analyzers		*sslyze*	*TLS Profiler*	*testssl.sh fork*	*TLS-Scanner*	*Immuniweb*	*Observatory*	*Inspector*	*TLSAssistant*
Guidelines	AgID								✓
	ANSSI								✓
	BSI				✓			✓	✓
	Mozilla	✓	✓				✓		✓
	NIST			✓	✓	✓			✓
Features	Open source	✓	✓	✓	✓		✓		✓
	Machine-readable report	✓		✓	✓				✓
	Checks with latest guidelines	✓	✓	✓	✓		✓	✓	✓
	Configuration file analyzer								✓
	Custom guidelines support								✓
	Actionable mitigations								✓

as a webapp, outputs a set of bullet points highlighting the differences between the current and the target Mozilla profile. The list is similar to the output generated by sslyze, but each divergent setting is shown separately from its class - e.g., even if cipher suites are usually configured in batch, TLS Profiles shows a bullet point for each one. Despite the difference in the output format, this tool shares the same limitations as sslyze.

- **testssl.sh:** is an open source command-line tool able to analyze a wide set of TLS vulnerabilities. An GitHub issue opened in March 2016 suggested the addition of a compliance check against NIST SP 800-52. The proposed changes are currently available in a fork of the tool created by David Cooper inside the branch “nist” (Cooper, 2021).
- **TLS-Scanner:** is a research tool developed by Ruhr-Universität Bochum to assist in evaluating TLS deploymentsS (Ruhr University Bochum and GmbH, 2023). With the release of version 4.2.0 in June 2022, TLS-Scanner gained the capability to compare a webserver configuration against NIST and BSI-issued guidelines, SP 800-52r2 and TR02102-2, respectively. At the time of writing, the default compliance analysis only consist of a counter that shows the number of passed, skipped and failed checks:

```
--|Guideline BSI TR-02102-2
Passed: 16
Skipped: 0
Failed: 8
Uncertain: 0

--|Guideline NIST SP 800-52r2
Passed: 24
Skipped: 5
Failed: 10
Uncertain: 0
```

Without a way to understand which tests have failed, it is impossible for a system administrator to understand which requirement has not been met and, consequently, how to fix it. However, it is possible to obtain the list of performed checks together with their results and related requirement level (e.g., `MUST`, `OPTIONAL`) by using the `ALL/DETAIL` reporting levels.

- **Immuniweb SSL Security Test:** is an online test suite (Immuniweb, 2024) able to perform, among other security-related analysis, PCI-DSS 3.2.1 (PCI, 2018) and NIST compliance checks. It reports if the target webserver has *major compliance issues* and labels each detected cipher and protocol, saying if it is part of a good configuration or a source of misconfiguration or weakness. It does not explain how to fix the detected misconfigurations, but it is more detailed than the analyzers described above.
- **Observatory** (Mozilla Foundation, 2024)**:** is a project by Mozilla designed to help system administrators to safely configure their deployments. In particular, the “TLS Observatory” tab exploits the Immuniweb engine and shows a summary containing the closeness to a Mozilla profile *Compatibility Level*, another name for the TLS profiles defined in (Mozilla Foundation, 2023). However, if the configuration widely differs from a profile, the tool returns “Non-compliant” as a result. This makes it impossible to understand how much the detected configuration actually differs from the target ones and, consequently, does not provide any hint on how to close the gap.
- **TLS Checklist Inspector** (achelos GmbH, 2023)**:** is an online analyzer that evaluates if an existing webserver is configured according to BSI TR-03116-4. Its output reports the entire list of

evaluated requirements and signals whether they are met or not. While it does not explain how to achieve compliance, its user interface is very clear and a potential checklist of the changes to be performed may be created after a quick glance.

6 CONCLUSIONS AND FUTURE WORK

During the process of configuring TLS, system administrators make several decisions to create a working deployment, decisions whose outcome is sometimes not obvious even to experts. To set a state-wide security threshold, many cybersecurity agencies issue technical guidelines to define the use and configuration of TLS. These guidelines can help to raise awareness and act as reference points, but understanding how to change an existing deployment can be difficult. In order to evaluate the existing compliance status of public administration webservers based in Europe and the United States, we conducted an initial study checking the compliance of 56 websites against their guidelines of reference.

The results of the study indicate that compliance with the requirements outlined in the guidelines is not straightforward; instead, it is notably affected by specific technological constraints, time constraints, and third-party modifications (e.g., updates to TLS libraries). The analysis was performed by employing a *Compliance Analyzer Module* built for TLSAssistant to mechanize the compliance analysis.

As future work, we plan to enhance the compliance analysis by providing more fine-grained actionable reports covering edge cases. For example, by adding contextual comments to inform system administrators that while a specific element may not guarantee strict compliance, certain exceptions may have security justification.

To further improve the *Compliance Analyzer Module* analysis capabilities, we also plan to expand the *TLS_Compl_Dset* by integrating technical requirements issued by other security agencies. Thanks to the GitHub deployment, this process can be crowdsourced and transparently audited by anybody.

We also plan to explore the possibility to create a custom language to formally define machine-readable requirements and to share the methodology with the relevant standards bodies – in particular the TLS working group in the IETF – to gather real-world feedback about its applicability and ways in which it can be used going forward.

Finally, we plan to extend the *Compliance Analyzer Module* to allow the generation of compliant TLS configurations starting with the simple selection of a guideline. Through this process, a system administrator will be able to produce a functional configuration file that can be used for the deployment of a webserver that directly adheres to regulatory standards.

ACKNOWLEDGEMENTS

This work has been partially supported by the joint laboratory between FBK and the Italian Government Printing Office and Mint, and by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU.

AVAILABILITY

All referenced materials are accessible via the companion website (Germenia et al., 2024).

REFERENCES

(2024). TLS Compliance Dataset. https://github.com/stfbk/tls-compliance-dataset.

Acar, Y., Backes, M., Fahl, S., Kim, D., Mazurek, M. L., and Stransky, C. (2016). You Get Where You're Looking for: The Impact of Information Sources on Code Security. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 289–305, San Jose. IEEE.

achelos GmbH (2023). TLS Checklist Inspector. https://www.tls-check.de/en.

Agenzia per l'Italia Digitale (2020). Raccomandazioni per l'adozione di TLS. https://cert-agid.gov.it/wp-content/uploads/2020/11/AgID-RACCSECTLS-01.pdf.

Agenzia per l'Italia Digitale (2024). Enti - Dataset - Opendata IPA. https://indicepa.gov.it/ipa-dati/dataset/enti. [Online; accessed 2024-03-04].

ANSSI (2020). Recommandations de sécurité relatives à TLS. https://www.ssi.gouv.fr/uploads/2017/07/anssi-guide-recommandations_de_securite_relatives_a_tls-v1.2.pdf.

Bradner, S. O. (1997). Key words for use in RFCs to Indicate Requirement Levels. RFC 2119.

Bundesamt für Sicherheit in der Informationstechnik (BSI) (2024). Technical Guideline TR-02102 Cryptographic Mechanisms. https://www.bsi.bund.de/EN/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/Technische-Richtlinien/TR-nach-Thema-sortiert/tr02102/tr02102_node.html.

Cooper, D. (2021). Test for conformance to NIST SP 800-52. https://github.com/drwetter/testssl.sh/issues/333.

Diquet, A. (2023). SSLyze. https://github.com/nabla-c0d3/sslyze.

Domestic Protocol Office of the Federal Government (2023). List of federal ministries. https://www.protokoll-inland.de/Webs/PI/EN/rank-and-titles/official-order/federal-ministries/list-of-federal-ministries-node.html. [Online; accessed 2024-03-04].

European Parliament (2016). Regulation (EU) 2016/679. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32016R0679.

Fett, D. (2023). TLS Profiler. https://github.com/danielfett/tlsprofiler.

Germenia, R., Manfredi, S., Rizzi, M., Sciarretta, G., Tomasi, A., and Ranise, S. (2024). Automating Compliance for Improving TLS Security Postures - An Assessment of Public Administration Endpoints. https://st.fbk.eu/complementary/SECRYPT2024.

Gorski, P. L., Iacono, L. L., Wermke, D., Stransky, C., Möller, S., Acar, Y., and Fahl, S. (2018). Developers Deserve Security Warnings, Too: On the Effect of Integrated Security Advice on Cryptographic API Misuse. In *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)*, pages 265–281, Baltimore, MD. USENIX Association.

Government of France (2024). Composition du Gouvernement | gouvernement.fr. https://www.gouvernement.fr/composition-du-gouvernement. [Online; accessed 2024-03-04].

IANA (2005a). Transport Layer Security (TLS) extensions. https://www.iana.org/assignments/tls-extensiontype-values/.

IANA (2005b). Transport Layer Security (TLS) parameters. https://www.iana.org/assignments/tls-parameters/.

Immuniweb (2024). SSL Security Test. https://www.immuniweb.com/ssl/.

Krombholz, K., Mayer, W., Schmiedecker, M., and Weippl, E. (2017). I Have No Idea What I'm Doing" - On the Usability of Deploying HTTPS. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1339–1356, Vancouver, BC. USENIX Association.

Li, F., Rogers, L., Mathur, A., Malkin, N., and Chetty, M. (2019). Keepers of the Machines: Examining How System Administrators Manage Software Updates For Multiple Machines. In *Fifteenth Symposium on Usable Privacy and Security (SOUPS 2019)*, Santa Clara, CA. USENIX Association.

Manfredi, S. (2023). Automated Assistance for Actionable Security: Security and Compliance of TLS Configurations. https://iris.unige.it/handle/11567/1112295.

Manfredi, S., Ceccato, M., Sciarretta, G., and Ranise, S. (2022). Empirical Validation on the Usability of Security Reports for Patching TLS Misconfigurations: User- and Case-Studies on Actionable Mitigations. In *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications 13*, pages 56–86.

Mozilla Foundation (2023). Security/Server Side TLS. https://wiki.mozilla.org/Security/Server_Side_TLS.

Mozilla Foundation (2024). Mozilla Observatory. https://observatory.mozilla.org. [Online; accessed 2024-03-05].

National Institute of Standards and Technology (NIST) (2019). Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-52r2.pdf.

Office of Management and Budget (2017). The HTTPS-Only Standard. https://https.cio.gov.

PCI (2018). Recommendation for Key Management, Part 1: General (Revision 3). https://docs-prv.pcisecuritystandards.org/PCI%20DSS/Standard/PCI_DSS_v3-2-1.pdf.

Ruhr University Bochum, P. U. and GmbH, H. (2023). TLS-Scanner. https://github.com/tls-attacker/TLS-Scanner.

Shodan Search Engine (2024). vuln:CVE-2014-0160. https://www.shodan.io/search?query=vuln%3ACVE-2014-0160. [Online; accessed 2024-03-06].

Synopsys, Inc (2014). The Heartbleed Bug. https://heartbleed.com.

Tiefenau, C., Häring, M., Krombholz, K., and von Zezschwitz, E. (2020). Security, Availability, and Multiple Information Sources: Exploring Update Behavior of System Administrators. In *Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020)*, pages 239–258. USENIX Association.

USAGov (2024). Department - USAGov Federal Directory Search in English Search Results. https://search.usa.gov/search?affiliate=usagov_en_az&query=Department. [Online; accessed 2024-03-04].