

# AuthGuide: Analyzing Security, Privacy and Usability Trade-offs in Multi-Factor Authentication

Davy Preuveneers, Sander Joos, and Wouter Joosen

imec-DistriNet, KU Leuven  
Celestijnenlaan 200A, B-3001 Heverlee, Belgium  
{davy.preuveneers,sander.joos,wouter.joosen}@kuleuven.be  
<https://distrinet.cs.kuleuven.be>

**Abstract.** Multi-factor authentication (MFA) reduces the risk of compromised credentials. However, selecting, configuring and combining different authentication factors is a challenge for both security administrators and end-users, as the configuration possibilities are large and the implications of choices on security, privacy and usability are not always well understood. This concern is further aggravated when the security administrator grants the end-user some flexibility for the selection of authentication factors, or when the latter are combined in a risk-adaptive manner. In this work, we present AUTHGUIDE, an authentication knowledge and configuration framework that increases the awareness about these trade-offs. Additionally, it raises the level of abstraction to configure MFA for a given identity and access management (IAM) platform through a series of questions by mapping the responses onto the IAM’s workflow of authentication steps for registration and login. We implemented AUTHGUIDE, validated it on top of the open source Keycloak IAM, and evaluated the effectiveness of our framework to analyze the security, privacy and usability trade-offs.

**Keywords:** Authentication · Security · Privacy · Usability.

## 1 Introduction

Two-factor authentication (2FA) and multi-factor authentication (MFA) [3,10] are effective measures to reduce the impact of breaches caused by stolen credentials and credential stuffing attacks. Yet, configuring an effective multi-factor or multi-modal authentication strategy remains a daunting task due to the non-trivial trade-offs between security, privacy and usability. For example, risk-based MFA solutions that use contextual factors—such as current and previous IP addresses, locations of the end-user, or browser fingerprints [1,8,4]—can help the relying party (RP) to quantify the risk and trigger additional step-up authentication actions. However, the same context factors are exploited for online tracking, and hence harm the privacy of the user [13]. Also, they may be rendered ineffective when web browsers implement countermeasures against such tracking.

Protocols and standards like FIDO2, WebAuthn and CTAP let web browsers authenticate users with public key cryptography, where the private key on the client is protected by a hardware security key or a mobile device implementing biometric authentication (e.g. fingerprint verification). Passwordless authentication sounds convenient, though the uptake of biometric authentication is slow. Previous research [11] on passwordless authentication has demonstrated that usability concerns remain. Furthermore, from a security point of view, the RP offering WebAuthn authentication, must trust the client—e.g. the biometric factor implementation on a mobile phone—used to unlock the private key. For example, the RP may not know the false positive rate (i.e. a security concern) and false negative rate (i.e. a usability concern) of each biometric factor on every mobile device. Furthermore, in 2019 the ‘Face Unlock’ feature of Google’s Pixel 4 was confirmed to work even when asleep<sup>1</sup>, and the use of gel-based screen protectors was also reported<sup>2</sup> to fool fingerprint authentication. Last but not least, end-users may understand the privacy benefits of their biometric templates never leaving their mobile device, but not necessarily the extent to which biometric factors can be subject to the above security threats.

When enabling multi-factor authentication in identity and access management (IAM) platforms, the number of configuration options offered to the security administrator is typically large, and the implications of the choices on security, privacy and usability are not always clear, also for end-users, hereby jeopardizing the onboarding of MFA. To address these challenges, we present AUTHGUIDE, an authentication knowledge framework that:

1. Embeds a body of knowledge to inform about the trade-offs of MFA
2. Analyzes the risk of the customization flexibility granted to the end-user
3. Raises the level of abstraction to simplify the configuration of MFA

AUTHGUIDE achieves this through a series of configuration questions with background and threat information on security, privacy and usability. It validates the responses against requirements from NIST SP 800-63B [5], and maps them onto an IAM’s workflow of mandatory, optional and alternative authentication steps. This workflow entails both the registration phase (e.g. the enrollment of security keys or OTP authentication factors) and the login phase. We implemented and validated our solution on top of Red Hat Keycloak, a state-of-practice and open source IAM platform. We evaluated the effectiveness of AUTHGUIDE to configure MFA, and its ability to analyze the security, privacy and usability trade-offs.

The remainder of this paper is structured as follows. Section 2 reviews relevant related work on multi-factor authentication strategies. The design and implementation of our solution is explained in section 3. The experimental evaluation and validation of AUTHGUIDE are discussed in section 4. We conclude with a summary of the main contributions of this work and a roadmap for further research in section 5.

<sup>1</sup> Google Pixel 4 Face Unlock works if eyes are shut (2019), <https://www.bbc.com/news/technology-50085630>

<sup>2</sup> Samsung: Anyone’s thumbprint can unlock Galaxy S10 phone (2019), <https://www.bbc.co.uk/news/technology-50080586>

## 2 Related work

In this section, we review relevant related work on multi-factor authentication, including adaptive and continuous authentication, to illustrate the complexity of understanding the security, privacy and usability trade-offs from the perspective of the different stakeholders, i.e. security administrators and end-users.

Dasgupta et al. [2] discussed adaptive multi-factor authentication strategies as a combination of the calculation of the trustworthiness of different authentication factors, and an adaptive strategy for selecting authentication factors based on their calculated trustworthiness, performance, surroundings and more. It combines a variety of biometric and non-biometric authentication factors, and also avoids repeated selections of the same set of factors in successive re-authentications to reduce the chance of establishing recognizable patterns. The solution was compared with the FIDO and Microsoft Azure MFA frameworks in a user study, and the proposed solution was found to be better. While the usability of the solution was evaluated, the perceived impact on the user’s privacy was not assessed. Wang et al. [14] analyzed 5 MFA solutions based on smart cards, passwords and biometrics, and they specifically investigated security failures of their deployment in multi-server environments under the assumption of various threat models (or adversary models). They found critical security and privacy issues in each of them, including vulnerabilities against stolen-verifier attacks, insider attacks, failing to provide forward secrecy, and the loss of user anonymity.

Many security and authentication guidelines—such as websites of governmental agencies<sup>3</sup>—strongly encourage the use of 2FA and MFA, though often only from an end-user perspective to recommend how to better protect online accounts. Other reports, such the NIST Special Publication 800-63B [5] on ‘Digital Identity Guidelines: Authentication and Lifecycle Management’ offer detailed technical requirements at different authenticator assurance levels, and with consideration of usability and privacy. While those reports are typically targeted towards security administrators, the latter have to consider not only the security, privacy and usability trade-offs of their MFA implementation, but also the degrees of freedom they are willing to offer to end-users to further customize the MFA experience to their personal preferences. Those trade-offs and their impact on the actual implementation and deployment are less straightforward.

Klieme et al. [7] presented FIDOnuous that builds upon the WebAuthn standard to support continuous authentication. While WebAuthn enables user-friendly passwordless authentication, as well as strong authentication methods with biometrics, it fails to detect an attack after a successful login. The authors propose a WebAuthn extension that uses an Android-based authenticator communicating over Bluetooth Low Energy (BLE), such that the relying party and the authenticator can continuously exchange authentication verifications. While the authors did not evaluate any specific continuous or behavioral authentication method, their simulation demonstrates the practical feasibility of the integration

---

<sup>3</sup> Safeonweb, Use two-factor authentication (2020), <https://www.safeonweb.be/en/use-two-factor-authentication>

with WebAuthn. From a privacy perspective, the risk assessment is carried out on the client, and no sensitive behavioral information is shared with the relying party. From a security point of view, the strength of the continuous authentication depends on the accuracy of the authentication methods used and their robustness against threats, such observation, spoofing and replay attacks by an active adversary.

Browser fingerprints are often considered in a risk-adaptive authentication strategy. Andriamilanto et al. [1] researched the adequacy of browser fingerprints as an authentication factor. These fingerprints were composed of 216 attributes, and the analysis was carried out on more than 4 million fingerprints. The authors investigated their distinctiveness and stability through time, as well as their collection time and size. Even though they concluded that browser fingerprints are a promising additional web authentication factor due to the unicity rate of 81 % for 1,989,365 browsers, caution is required. Their own analysis indicates that the unicity for mobile fingerprints is 39.9 % and far lower than the 88.4 % for desktop fingerprints. This observation confirms previous results by Spooen et al. [12]. Also, the impact of countermeasures against tracking, such as FP-Block [13], is not discussed. Laperdrix et al. [8] investigated canvas fingerprinting, a subset of browser fingerprinting, as a user-friendly authentication factor. Their solution is not vulnerable to replay attacks due to its parameterization with a challenge/response protocol. They investigated more than 1.1 million fingerprints and found that the technique is sufficiently deterministic for verification even in the presence of some canvas poisoners that add noise to canvas elements as means to mitigate tracking. Nonetheless, the authors consider the option for users to whitelist their solution intended for authentication. As such, they conclude that canvas fingerprinting is a suitable mechanism. From a privacy perspective, the authors argued that browser fingerprinting is intrusive due to its ability to link user visits, a privacy concern that was already raised earlier by Eckersley in the Panoptick project [4]. However, the proposed method is intended for first-party websites that already use first-party cookies to track users, and as such does not impose any additional linkability threats.

Karegar et al. [6] studied user perceptions on the widely deployed fingerprint recognition on smartphones, often used to unlock the device or to authenticate against remote applications. More specifically, they investigated in an online survey how 100 individuals think that fingerprint recognition works and this in contrast to PIN codes, as well as privacy and possible other issues with this biometric authentication factor. They compared the attitudes of users and non-users. Their user study demonstrated amongst others that even participants reporting a higher level of knowledge in security do not necessarily have a good perception about access to fingerprint patterns and PIN codes of mobile apps.

### 3 AuthGuide: design and implementation

The main use case of AUTHGUIDE is security administrators configuring their IAM platforms by mapping individual options in AUTHGUIDE onto a specific

IAM workflow of authentication steps for registration and login. To configure MFA for different platforms, AUTHGUIDE generates a custom specialized script to be executed by the security administrator. Additionally, AUTHGUIDE provides security administrators and end-users a breakdown of various security, privacy and usability requirements and trade-offs. As such, the goal of AUTHGUIDE is not to improve any particular authentication factor, but rather to analyze (1) the security, privacy and usability implications of different authentication factors, (2) their combination in an MFA configuration, and (3) the consequences of granting some flexibility on authentication factor selection to the end-user. Our solution builds upon the NIST set of technical requirements [5] to evaluate the assurance level of MFA implementations, as well as their impact on privacy and usability. It validates the configuration options of the security administrator with respect to the ‘SHALL’ and ‘SHOULD’ requirement notations and conventions (including the negative forms), the degrees of freedom for customization granted to the end-user, as well as influences of external elements beyond control of the security administrator of an IAM and/or end-user.

### 3.1 Modeling the configuration space of authentication factors

AUTHGUIDE models the configuration space of a variety of knowledge, possession, inherence, contextual and behavioral authentication factors, and exposes this body of knowledge to the security administrator in the form of an online configuration wizard. The configuration options can be set by the IAM’s security administrator, and optionally further customized by the end-user according to personal preferences and the availability of the necessary equipment.

Even for relatively simple authentication factors such as passwords, the security administrator is typically faced with several configuration options that influence the security and usability trade-off. Some of them are listed below:

- *Are passwords a mandatory or an optional authentication factor?*  
TRADE-OFF: Passwords do not need dedicated hardware, but may be reused.
- *What is the minimal length and complexity of a password?*  
TRADE-OFF: Entering long and strong passwords harms the user experience.
- *Does the implementation offer a password strength meter?*  
TRADE-OFF: Users may not be able to reasonably estimate the relative strength of different passwords.
- *Does the implementation offer to display the secret password?*  
TRADE-OFF: Displaying the password may simplify entering the correct password, but make shoulder surfing attacks easier to carry out.
- *What is the maximum limit of failed authentication attempts?*  
TRADE-OFF: A lower limit reduces the security risk, but also the number of attempts to remember and enter a rarely used password.
- *How often should passwords be changed?*  
TRADE-OFF: Regularly changing passwords improves security, but increases the mental burden to remember ever-changing passwords.

<b>Authenticator Type</b>	<b>SHALL</b>	<b>SHOULD</b>
Memorized Secrets	24	15
Look-Up Secrets	14	0
Out-of-Band Devices	27	4
Single-Factor OTP Device	11	1
Multi-Factor OTP Devices	20	1
Single-Factor Cryptographic Software	3	1
Single-Factor Cryptographic Devices	9	1
Multi-Factor Cryptographic Software	8	2
Multi-Factor Cryptographic Devices	8	1

**Table 1.** Amount of requirements (including negative form) by authenticator type.

<b>General Authenticator Requirements</b>	<b>SHALL</b>	<b>SHOULD</b>
Physical Authenticators	2	0
Rate Limiting	2	1
Use of Biometrics	14	2
Attestation	11	1
Verifier Impersonation Resistance	6	0
Verifier-CSP Communications	1	0
Verifier-Compromise Resistance	5	0
Replay Resistance	0	0
Authentication Intent	1	0
Restricted Authenticators	2	0

**Table 2.** Amount of general requirements (including negative form) for authenticators.

The first configuration option is one that can be delegated to the end-users to enable a passwordless authentication experience for those with alternative means of authentication. Others can be decided upon by the security administrator, or be constrained by the underlying authentication platform (e.g. support for the WebAuthn standard) or the availability of certain hardware (e.g. One-Time Password devices). Even if an IAM platform offers the above capabilities, the NIST SP 800-63B [5] guidelines state, for example, that verifiers **SHOULD NOT** require memorized secrets—such as passwords—to be changed arbitrarily or periodically, but only if there is evidence of compromise of the authenticator.

Note that certain end-user choices can have a positive or negative impact on security and usability over which the security administrator or the MFA implementation have limited control. Examples are the ability of end-users to store their passwords (in an unprotected document versus in a password manager), or the reuse of previous passwords that unknowingly to the end-user and security administrator may have already been compromised<sup>4</sup>. Also security tools like password managers can raise privacy concerns due to the presence of trackers<sup>5</sup>.

<sup>4</sup> Have I Been Pwned?, <https://haveibeenpwned.com/>

<sup>5</sup> Exodus Privacy: LastPass 4.11.18.6150 has 7 trackers (Mar 2021), <https://reports.exodus-privacy.eu.org/en/reports/165465/>

Table 1 gives an overview of the amount SHALL/SHOULD requirements, including their negative form, for a subset of authenticator types in terms of how often they occur in the NIST SP 800-63B guidelines. The column SHALL indicates the number of requirements to be followed strictly, whereas SHOULD counts the number of preferred but not necessarily required courses of action. We excluded the CAN and MAY requirements in the table as they are less stringent (and also less frequent in the guidelines). Table 2 lists in a similar manner the amount of general requirements for authenticators. From the above two tables, we discard those implementation specific requirements without any configuration option. An example of such a requirement for password authenticators is the fact that the salt SHALL be at least 32 bits in length and be chosen arbitrarily. Such requirements should be validated directly against the IAM implementations or specific biometric authentication factors to possibly prohibit their usage. Of the 198 requirements in Tables 1 and 2, AUTHGUIDE discards 125 requirements that are either irrelevant for the IAM itself (e.g. those targeting end-user devices) or IAM implementation specific (e.g. use of approved cryptography methods). The latter can be validated separately if the IAM feature is non-configurable, or the script generated by AUTHGUIDE directly configures the IAM to meet the requirements. The actual number of requirements being validated depends on the remaining configuration options selected in AUTHGUIDE. For example, a password-less authentication configuration would not check the password-related requirements (minimum length, acceptable characters, etc.). Other requirements are conditionally dependent on the desired authenticator assurance level (the NIST guidelines define 3 levels).

The remaining configuration options for the security administrator and/or end-user are evaluated by a Drools rule engine<sup>6</sup> in AUTHGUIDE. Based on the responses in AUTHGUIDE the list of requirements is further narrowed down to consider only those that are relevant for the selected set of authentication factors. AUTHGUIDE then evaluates (1) the number of relevant requirements and the amount of violations, (2) a base aggregated score for the SHALL and SHOULD requirements, and (3) an upper-bound and lower-bound to account for those options granted to the end-user for customization.

### 3.2 Registration and replacement of authentication factors

For a given MFA configuration, an authentication factor can be registered or bound to the account of the end-user during enrollment, or later when the end-user adds an acceptable authenticator to strengthen the security of the account.

For each authentication factor, there should be a backup and recovery strategy in case the authenticator is not available, lost, damaged, stolen or compromised due to a data breach. These recovery mechanisms need to be secure and user-friendly as well. For example, SMS and push notifications on a mobile have been deprecated as authentication factors due to the possibility of phishing or

---

<sup>6</sup> <https://www.drools.org>

SIM swapping attacks [9] or interception by IMSI-catchers. Therefore, recovery strategies should not rely on them either.

The consequences on the MFA workflow (i.e. the order of different mandatory and alternative authentication steps) are two-fold: (1) not every end-user will have the same MFA configuration and the workflow should be able to adapt to that, and (2) the workflow should support escalation to alternative authentication factors. For example, end-users that opted for a passwordless authentication experience should not be shown a form to fill in both their username and password, but rather only the username.

The screenshot shows the AuthGuide wizard interface. At the top, the logo 'DistriNet' is on the left, and 'Home Links' is on the right. The main heading is 'AuthGuide: A wizard for Multi-Factor Authentication' by Davy Preuveneers - imec-DistriNet, KU Leuven. Below this is a navigation bar with 'Wizard', 'Analysis', and 'Export' tabs. A progress bar shows seven steps: Generic (2FA and MFA), Knowledge (Password, PIN cod...), Possession (OTP, smart card, ...), Inherence (Biometric factors), Context (Time, location, risk, ...), Lifecycle (Binding, renewal, ...), and Finish (Complete wizard). The 'Knowledge' step is currently active.

**Available memorized secrets** [AuthGuide Info](#)

- Password or passphrase
- PIN code
- Swipe pattern
- Answer to secret question
- Other knowledge factor:

**Number of memorized secrets required to authenticate?**

(Less secure, more convenient) 0 1 2 3 (More secure, less convenient)

**Password authentication factor is mandatory?** [AuthGuide Info](#)

No  Yes

**Minimum length of a password?**

(Less secure, more convenient) 5 6 7 8 9 10 11 12 13 14 15 (More secure, less convenient)

**Maximum length of a password?**

100

**Acceptable characters for passwords**

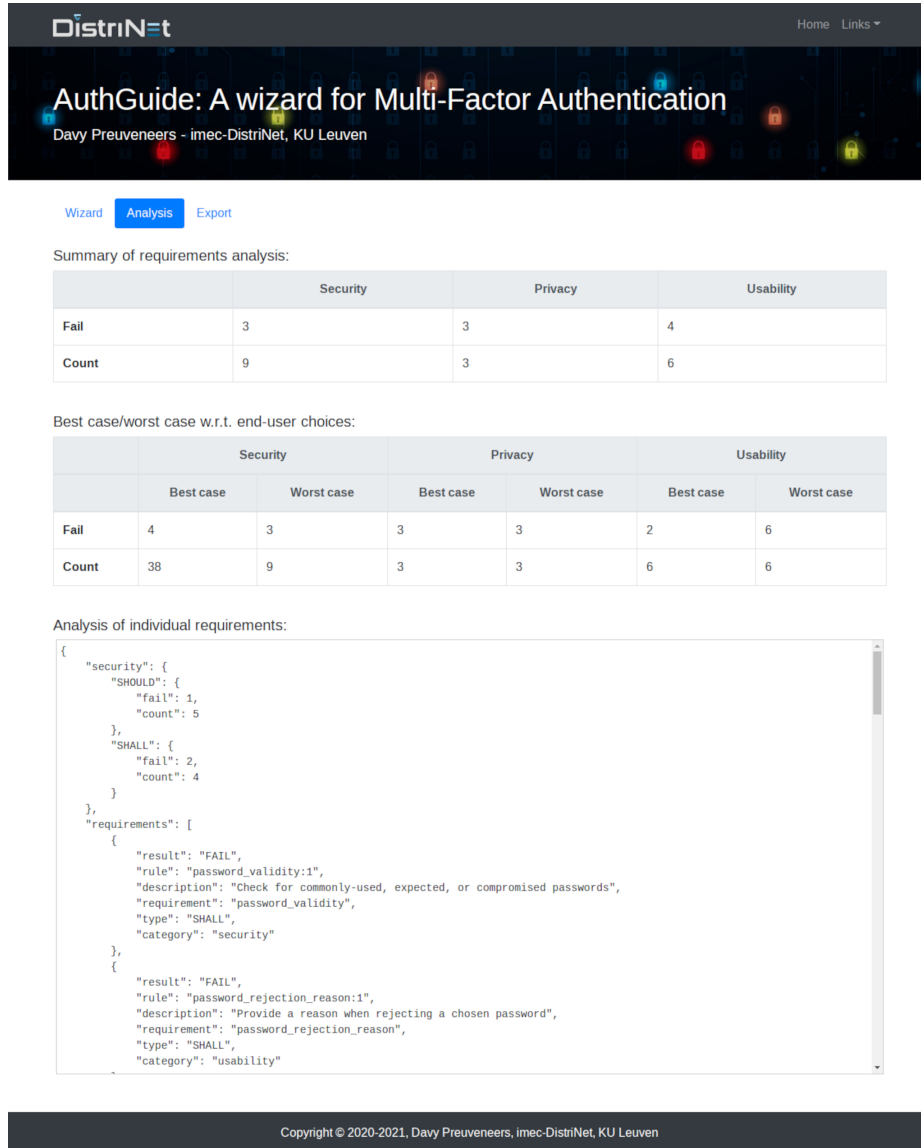
- Alphanumeric characters
- Space character
- Printable ASCII characters
- Unicode characters

**Enable password hints?**

No  Yes

**Fig. 1.** The AUTHGUIDE wizard for Multi-Factor Authentication configuration and requirement validation.





The screenshot shows the AuthGuide dashboard interface. At the top, there is a navigation bar with the 'DistriNet' logo and 'Home Links' menu. The main header reads 'AuthGuide: A wizard for Multi-Factor Authentication' by Davy Preuveneers at imec-DistriNet, KU Leuven. Below this, there are three tabs: 'Wizard', 'Analysis' (which is active), and 'Export'.

The 'Analysis' section is titled 'Summary of requirements analysis:' and contains a table with the following data:

	Security	Privacy	Usability
Fail	3	3	4
Count	9	3	6

Below this is the 'Best case/worst case w.r.t. end-user choices:' section, which contains a more detailed table:

	Security		Privacy		Usability	
	Best case	Worst case	Best case	Worst case	Best case	Worst case
Fail	4	3	3	3	2	6
Count	38	9	3	3	6	6

The 'Analysis of individual requirements:' section displays a JSON object with the following structure:

```
{
  "security": {
    "SHOULD": {
      "fail": 1,
      "count": 5
    },
    "SHALL": {
      "fail": 2,
      "count": 4
    }
  },
  "requirements": [
    {
      "result": "FAIL",
      "rule": "password_validity:1",
      "description": "Check for commonly-used, expected, or compromised passwords",
      "requirement": "password_validity",
      "type": "SHALL",
      "category": "security"
    },
    {
      "result": "FAIL",
      "rule": "password_rejection_reason:1",
      "description": "Provide a reason when rejecting a chosen password",
      "requirement": "password_rejection_reason",
      "type": "SHALL",
      "category": "usability"
    }
  ]
}
```

At the bottom of the dashboard, there is a copyright notice: 'Copyright © 2020-2021, Davy Preuveneers, imec-DistriNet, KU Leuven'.

**Fig. 2.** The AUTHGUIDE requirement validation summary.

### 3.3 AuthGuide implementation

AUTHGUIDE is implemented as an HTML5 dashboard on top of a backend implemented with the Java-based Spring MVC framework. The latter maintains the MFA knowledge base of the wizard to inform the security administrator

or end-user about security, privacy and usability concerns. Fig. 1 depicts the web-based wizard interface to configure Multi-Factor Authentication configuration and carry out the security, privacy and usability trade-off analysis. The ‘AuthGuide Info’ button triggers a pop-up window that offers the security administrator or end-user additional background information on possible known threats and trade-offs.

AUTHGUIDE manages the list of configuration options per authentication factor, the mapping onto SHALL/SHOULD requirements, and any dependencies across the choices and requirements. Currently, it manages 73 configuration options in the wizard that are mapped onto a subset of the SHALL/SHOULD requirements of Tables 1 and 2, and that are conditionally exposed in the wizard depending on previously selected options. The requirements are implemented as a ruleset evaluated by the Drools 7.54 rule engine (see Fig. 3 in Appendix for examples of the ruleset). The main reason to rely on a rule engine rather than hardcoding a set of if-then-else rules in the application is to simplify the implementation and evaluation of conditional requirements and dependencies across requirements. Additionally, by managing these rules external to the application, they can be easily updated whenever new recommendations are proposed. As shown in Fig. 2, the outcome of the analysis is a report listing:

- The name of the requirement
- The type of the requirement (i.e. SHALL or SHOULD, or the negative form)
- The outcome of the validation (i.e. PASS or FAIL, and an explanation)
- The name of the Drools rule that produced this result

It also checks for violations against the 3 authentication assurance levels in NIST Special Publication 800-63B [5], though it is not a full compliance analysis tool as AUTHGUIDE does not verify the implementation-specific requirements listed in this report. If certain configuration options can be customized by the end-user, AUTHGUIDE evaluates both a best-case and a worst-case configuration scenario, as depicted in Fig. 2. AUTHGUIDE as such can not only inform the user about security-privacy-usability trade-offs of individual configuration options, but also about trade-offs for configurations as a whole.

The last feature of AUTHGUIDE is its ability to translate the JSON configuration file produced by the wizard into a configuration shell script for state-of-practice IAM platforms (currently only tested with Red Hat’s Keycloak 12.0.4) to simplify the configuration of MFA. The remaining customization left for the security administrator before executing this script is to complete the URL end-point of the IAM and the name of the realm for which MFA needs to be configured. After completing these deployment specific details, the security administrator can execute the configuration script to create or update the realm.

## 4 Evaluation

To evaluate the performance and practical feasibility of AUTHGUIDE, we defined four different MFA scenarios:

1. **Fixed single factor authentication:** This scenario allows for single factor authentication, using as default the password authenticator. This is a typical scenario for sites that do not yet support MFA.
2. **Flexible single factor authentication:** This scenario augments the previous by allowing the end-user to replace the password authentication factor to enable a password-less experience (e.g. token or biometric) after enrollment.
3. **Mandatory 2FA with fixed authentication factors:** The scenario enforces 2FA during enrollment of the end-user with the two authentication factors fixed by the security administrator, i.e. a password combined with a Time-based One-Time Password (TOTP).
4. **Mandatory MFA with flexible authentication factors:** This scenario supports authentication with multiple knowledge, possession, inherence and context factors. The end-user can customize the configuration and select two or more factors, but they must belong to different categories.

Each of the above scenarios allows for multiple configurations, for example, by changing the valid password constraints or the password expiry policy, a 6 or 8 digit TOTP, the number and type of authentication factors in the mandatory MFA scenario, etc. For each scenario, we produced 3 different variants, resulting in 12 variants in total.

#### 4.1 Performance evaluation

To validate AUTHGUIDE, we first evaluated the efficiency of validating the SHALL and SHOULD requirements. We tested each of the 12 variants 10 times in random order on a machine with an Intel Core i7-7700U CPU running at 3.60GHz, and with 32GB of memory. The Spring Boot 2.4.6 application with the embedded Drools 7.54 rule engine runs with OpenJDK 11.0.11 in a Ubuntu 21.04 Linux environment.

We measured the time the Java application needs for the rule-based analysis, i.e. excluding the time to complete the AUTHGUIDE wizard. Overall, the time to evaluate the Drools rules to check the requirements varies between 0.609 and 2.149 milliseconds, which is well below our pre-defined target of 0.1 second. Additionally, the memory used by the Java application (i.e. the maximum resident set size (RSS)) is 678528 kbytes. Based on these results, we believe it would be practically feasible to carry out the requirements validation directly within the browser with support of a proper JavaScript-based rule engine. However, to run AUTHGUIDE's application logic completely in the browser, the code generation for the IAM's configuration script also needs to be refactored and reimplemented.

#### 4.2 Configuration support for the security administrator

For a variant of each of the above 4 scenarios, we compared the time required to configure a realm within a Keycloak 12.0.4 deployment:

1. Manually through the configuration dashboard of Keycloak

Scenario	Keycloak Dashboard	AuthGuide Dashboard	AuthGuide Script
Scenario 1	1 min 36 s	35 s	20 s
Scenario 2	2 min 48 s	41 s	23 s
Scenario 3	3 min 51 s	45 s	35 s
Scenario 4	4 min 23 s	59 s	43 s

**Table 3.** Time to configure 4 single-factor and multi-factor authentication scenarios.

## 2. Using the shell configuration script produced by AUTHGUIDE

For the second approach, the Keycloak’s ‘kcadm’ command-line utility is already configured in advance. Both approaches were executed by an experienced Keycloak user. Additionally, the authentication flows to be manually configured in Keycloak for each of the four authentication variants were defined in advance. The rationale for this decision is the desire to rule out any influences caused by the need to correct misconfigured authentication flows.

The results of this experiment are shown in Table 3. One should compare the time required to configure a variant with the Keycloak Dashboard versus the combined time to achieve the same with the AUTHGUIDE Dashboard and Script. The time for AUTHGUIDE Script is the time required for the security administrator to execute the generated script as AUTHGUIDE does not directly update Keycloak. The time required is less for AUTHGUIDE, and more outspoken for the more sophisticated authentication flows.

### 4.3 Analysis of security, privacy and usability trade-off

We evaluated to what extent the different security, privacy and usability requirements in NIST SP 800-63B [5] have been addressed. The result is (a) an indication of the authenticator assurance level achieved, and (b) the number of SHALL and SHOULD requirements that have been met relative to the number of requirements that were assessed (see Fig. 2). The trade-off analysis offers 6 values subdivided by:

- **Category:** Security, privacy and usability
- **Upper- and lower-bound:** Best- and worst-case w.r.t. end-user choices

Within the frame of this research, we did not quantitatively evaluate the level of increased awareness about MFA threats and trade-offs while using AUTHGUIDE, nor did we investigate MFA configuration mistakes when directly using the IAM’s configuration interface and AUTHGUIDE’s ability to avoid them. This assessment involving stakeholders with different levels of IAM expertise and MFA experience will be part of a future user study.

## 5 Conclusion

The contribution of this work is AUTHGUIDE, an authentication knowledge and configuration framework. It aims to increase the awareness about security, pri-

vacancy and usability trade-offs by analyzing to what degree relevant requirements of NIST SP 800-63B [5] have been addressed, while also considering the implications of granting some flexibility on authentication factor selection to the end-user. For the security administrator, it simplifies the process of configuring MFA for a given identity and access management (IAM) platform. Our experimental evaluation demonstrated the practical feasibility and the added benefit of AUTHGUIDE for the security administrator. A user study evaluating the increased awareness about the aforementioned trade-offs and the ability to avoid MFA configuration mistakes is pending.

As future work, we aim for a more balanced weighing of the different types of requirements, as well as enhanced configuration support for alternative IAM platforms with plug-in support for the validation of implementation specific requirements.

## Acknowledgments

This research is partially funded by the Research Fund KU Leuven and by the Flemish Government’s Cybersecurity Initiative Flanders. Work for this paper was supported by the European Commission through the H2020 project CyberSec4Europe (<https://www.cybersec4europe.eu/>) under grant No. 830929.

## References

1. Andriamilanto, N., Allard, T., Guelvouit, G.L.: “guess who?” large-scale data-centric study of the adequacy of browser fingerprints for web authentication. In: Barolli, L., Ponziszewska-Maranda, A., Park, H. (eds.) *Innovative Mobile and Internet Services in Ubiquitous Computing*. pp. 161–172. Springer International Publishing, Cham (2021)
2. Dasgupta, D., Roy, A., Nag, A.: Toward the design of adaptive selection strategies for multi-factor authentication. *Computers & Security* **63**, 85–116 (2016). <https://doi.org/https://doi.org/10.1016/j.cose.2016.09.004>, <https://www.sciencedirect.com/science/article/pii/S016740481630102X>
3. Dasgupta, D., Roy, A., Nag, A.: *Multi-Factor Authentication*, pp. 185–233. Springer International Publishing, Cham (2017). [https://doi.org/10.1007/978-3-319-58808-7\\_5](https://doi.org/10.1007/978-3-319-58808-7_5), [https://doi.org/10.1007/978-3-319-58808-7\\_5](https://doi.org/10.1007/978-3-319-58808-7_5)
4. Eckersley, P.: How unique is your web browser? In: Atallah, M.J., Hopper, N.J. (eds.) *Privacy Enhancing Technologies*. pp. 1–18. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
5. Grassi, P., Perlner, R., Newton, E., Regenscheid, A., Burr, W., Richer, J., Lefkovitz, N., Danker, J., Theofanos, M.: *Digital identity guidelines: Authentication and lifecycle management [including updates as of 03-02-2020]* (2017-12-01 2017). <https://doi.org/https://doi.org/10.6028/NIST.SP.800-63b>
6. Karegar, F., Pettersson, J.S., Fischer-Hübner, S.: Fingerprint recognition on mobile devices: Widely deployed, rarely understood. In: Doerr, S., Fischer, M., Schrittwieser, S., Herrmann, D. (eds.) *Proceedings of the 13th International Conference on Availability, Reliability and Security, ARES 2018, Hamburg, Germany, August 27-30, 2018*. pp. 39:1–39:9. ACM (2018).

- <https://doi.org/10.1145/3230833.3234514>, <https://doi.org/10.1145/3230833.3234514>
7. Klieme, E., Wilke, J., van Dornick, N., Meinel, C.: Fido2/webauthn extension to support continuous web authentication. In: 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). pp. 1857–1867 (2020). <https://doi.org/10.1109/TrustCom50675.2020.00254>
  8. Laperdrix, P., Avoine, G., Baudry, B., Nikiforakis, N.: Morellian analysis for browsers: Making web authentication stronger with canvas fingerprinting. In: Perdisci, R., Maurice, C., Giacinto, G., Almgren, M. (eds.) *Detection of Intrusions and Malware, and Vulnerability Assessment*. pp. 43–66. Springer International Publishing, Cham (2019)
  9. Lee, K., Kaiser, B., Mayer, J., Narayanan, A.: An empirical study of wireless carrier authentication for sim swaps. USENIX Association, Virtual Conference (08/2020 2020), <https://www.usenix.org/system/files/soups2020-lee.pdf>
  10. Ometov, A., Bezzateev, S., Mäkitalo, N., Andreev, S., Mikkonen, T., Koucheryavy, Y.: Multi-factor authentication: A survey. *Cryptography* **2**(1) (2018). <https://doi.org/10.3390/cryptography2010001>, <https://www.mdpi.com/2410-387X/2/1/1>
  11. Oogami, W., Gomi, H., Yamaguchi, S., Yamanaka, S., Higurashi, T.: Observation study on usability challenges for fingerprint authentication using webauthn-enabled android smartphones. In: *Symposium on Usable Privacy and Security (SOUPS 2020)*. USENIX Association (Aug 2020)
  12. Spooren, J., Preuveneers, D., Joosen, W.: Mobile device fingerprinting considered harmful for risk-based authentication. In: *Proceedings of the Eighth European Workshop on System Security. EuroSec '15*, Association for Computing Machinery, New York, NY, USA (2015). <https://doi.org/10.1145/2751323.2751329>, <https://doi.org/10.1145/2751323.2751329>
  13. Torres, C.F., Jonker, H., Mauw, S.: Fp-block: Usable web privacy by controlling browser fingerprinting. In: Pernul, G., Y A Ryan, P., Weippl, E. (eds.) *Computer Security – ESORICS 2015*. pp. 3–19. Springer International Publishing, Cham (2015)
  14. Wang, D., Zhang, X., Zhang, Z., Wang, P.: Understanding security failures of multi-factor authentication schemes for multi-server environments. *Computers & Security* **88**, 101619 (2020). <https://doi.org/https://doi.org/10.1016/j.cose.2019.101619>, <https://www.sciencedirect.com/science/article/pii/S016740481930166X>

## Appendix

---

```

1 // 8 characters in length
2 rule "password_min_length:1"
3 activation-group "password_min_length"
4 salience 500
5 when
6   MFARequirement($r: req("password_min_length"), $r != null)
7   $config: MFAConfig(
8     "password" memberOf get("knowledge_factor") &&
9     get("password_min_length") < 8
10  )
11   $validate: MFAValidate()
12 then
13   $validate.add(drools.getRule().getName(), "Password too short",
14     MFAValidate.FAIL, $r);
15 end
16
17 rule "password_min_length:2"
18 activation-group "password_min_length"
19 salience 100
20 when
21   MFARequirement($r: req("password_min_length"), $r != null)
22   $config: MFAConfig()
23   $validate: MFAValidate()
24 then
25   $validate.add(drools.getRule().getName(), "Success",
26     MFAValidate.PASS, $r);
27 end
28
29 /*****/
30
31 // Verifiers SHOULD NOT require memorized secrets to be changed
32 // arbitrarily (e.g., periodically)
33 rule "password_expiry:1"
34 activation-group "password_expiry"
35 salience 500
36 when
37   MFARequirement($r: req("password_expiry"), $r != null)
38   $config: MFAConfig(
39     "password" memberOf get("knowledge_factor") &&
40     get("password_expiry") > 0
41   )
42   $validate: MFAValidate()
43 then
44   $validate.add(drools.getRule().getName(), "Do not set a password
45     expiry policy", MFAValidate.FAIL, $r);
46 end
47
48 rule "password_expiry:2"
49 activation-group "password_expiry"
50 salience 100
51 when
52   MFARequirement($r: req("password_expiry"), $r != null)
53   $config: MFAConfig()
54   $validate: MFAValidate()
55 then
56   $validate.add(drools.getRule().getName(), "Success",
57     MFAValidate.PASS, $r);
58 end

```

---

**Fig. 3.** Analyzing MFA requirements with a Drools ruleset, illustrating an example 'SHALL' and 'SHOULD' requirement for passwords from NIST SP 800-63B.