Risk-Based Authenticator for Web Applications

ROLAND H. STEINEGGER, DANIEL DECKERS, PASCAL GIESSLER, AND SEBASTIAN ABECK,
Cooperation & Management, Karlsruhe Institute of Technology (KIT)

Web applications for consumers often require authenticated users in order to offer their services. In this context, consumers expect authentication to be easy to use and their resources to be secured properly. But, authentication in web applications is often vulnerable, e.g., passwords can be stolen, fingerprints can be imitated or the authentication protocol implementation may have a security flaw. Several best practices solving this problem evolved in several web applications. We analyzed such solutions that continuously and transparently collect data on the user to learn their typical behavior and detect anomalies. Based on this analysis, we describe the security pattern risk-based authenticator and exemplify its application in the SmartCampus, a service-oriented web application.

CCS Concepts: •**Security and privacy** → **Authentication;** •**Software and its engineering** → *Design patterns;*

Additional Key Words and Phrases: Security Pattern, Risk, Authentication, Consumer Identity and Access Management, Web Applications

ACM Reference Format:

Roland H. Steinegger, Daniel Deckers, Pascal Giessler, and Sebastian Abeck. 2016. Risk-Based Authenticator for Web Applications. In *Proceedings of 21st European Conference on Pattern Languages of Programs (EuroPLoP 2016)*. ACM Press, New York, NY, 11 pages.
DOI: http://dx.doi.org/10.1145/3011784.3011800

1. INTRODUCTION

Every web application faces the recurring problem of authenticating users and controlling access on resources by enforcing the authorization. These problems can be assigned to the area of Identity and Access Management (IAM) . In the past decades, IAM was mainly influenced by problems within enterprises, called Enterprise IAM (EIAM). This changed in the past years. Enterprises now start offering their services to consumers through web applications, which lead to the field of Consumer IAM (CIAM).

In web applications for consumers, the IAM may work as enabler for success because of its key features [McQuaide 2003]. As enabler, consumer web applications need to be easy to use, otherwise the consumer moves to a competitor. On the one hand this pressure within CIAM leads to innovative solutions, but on the other hand security measures offering a high security level are not used. For example, password authentication is still widely used for verifying the user's authenticity, although it is known to be insecure. At least since 1989, weaknesses of password authentication are discussed [Jobusch and Oldehoeft 1989] and the discussion continues until today [Everett 2016]. But apart from that, authentication is generally vulnerable. Either the mechanism or its implementation is vulnerable, e.g. fingerprints can be imitated [Paul and Irvine 2016] and foreign components may come with vulnerabilities [OWASP Foundation 2014], or the application using the

Author's address: R. H. Steinegger, D. Deckers, P. Giessler and S. Abeck, Zirkel 2, 76137 Karlsruhe; email: {steinegger, giessler, abeck}@kit.edu, daniel.deckers@ic-consult.com;

EuroPLoP '16, July 06 - 10, 2016, Kaufbeuren, Germany
© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM. ACM 978-1-4503-4074-8/16/07...$15.00
DOI: http://dx.doi.org/10.1145/3011784.3011800

authentication mechanism and handling the user's session has vulnerabilities, which is in the top ten security problems of web applications [OWASP Foundation 2015].

This leads to the problem, that the term authentic gets (more) fuzzy; a user is authentic just up to a certain percentage. Tackling this issue, enterprises developed solutions that continuously verify the user's authenticity and take measures, if there is doubt. For example, access to a web-mail application is declined, if accessing it from a foreign country, or users are not able to see their bank balance until entering a one-time-password, that is sent to their mobile phone. Ready to use software is available to support such scenarios [PistolStar Inc. 2016; SafeNet Inc. 2016; CA Technologies 2016; ForgeRock Community 2016; NuDetect Solutions 2016]. Many web application , but we could not find a (security) pattern describing it.

Thus, we use information from several sources and introduce the Risk-Based Authenticator (RBA) pattern in section 4. The security pattern addresses software architects and developers building web applications for consumers. It introduces a structure separating the concerns of a risk-based authentication. The pattern evolved by implementing a risk-based authentication for the SmartCampus web application at the Karlsruhe Institute of Technology (KIT) including several discussions with experts from an IAM consulting company. Additionally, a shepherding with Eduardo Fernandez-Buglioni as well as writers workshops on the European Conference on Pattern Languages of Programs (EuroPLoP) helped to greatly improve the pattern.

2. SECURITY PATTERNS REGARDING AUTHENTICATION

In this section, we give the foundation for the RBA. An introduction into the IAM shows the field in which the RBA is embedded. The RBA is described as security pattern, therefore, we briefly present this concept and show patterns that are closely connected to the RBA. Additionally, other work related to the RBA and its usability improvement is discussed.

2.1 Security Patterns

Security Patterns are a tool to conserve knowledge of security experts. The idea of design patterns is based on the concept of patterns in architecture [Alexander 1977]. Each security pattern provides a solution for a security problem in a specific context [Schumacher et al. 2006]. They can be described with the same structure as design patterns. Security patterns always appear in relationship with other patterns [VanHilst and Fernandez-Buglioni 2007]. Focusing on a specific domain, patterns can be organized in pattern languages or pattern systems in order to describe relations between them.

The IAM is such a domain, which aims at the problem that the right individual gets access to the right resources to the right time [Gartner Inc. 2016]. Access management solves the problem of authentication and authorization, where authentication means the proved identification of an individual. Further, solutions in the IAM can be divided into solutions for classical Enterprise IAM (EIAM) and Consumer IAM (CIAM).

EIAM deals with problems arising in enterprises, where employees or partners need access to applications of the company. These applications are often placed in the enterprises intranet, where they are shielded by a firewall. IAM problems that arise, when these enterprises offer services to customers through web applications, are part of the CIAM. Usability is a competitive advantage in CIAM, whereas security is more important in EIAM [McQuaide 2003].

2.2 Authentication Patterns

The pattern language for identity management [Delessy et al. 2007] as well as the the abstract view on the authenticator pattern [Fernandez-Buglioni et al. 2014] describe the relationship of the most relevant security patterns for the RBA. A short overview on these patterns is given next.

While security patterns for every software development phase exist, we focus on security patterns for building secure architectures. In Fernandez-Buglioni [2013] the patterns are aligned according to the archi-

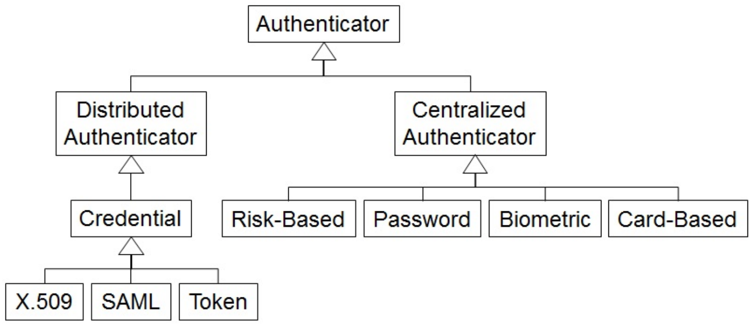

Fig. 1. Aligning the RBA within the patterns of [Fernandez-Buglioni et al. 2014]

tectural level and concern. The risk-based authenticator, we introduce in this work, can be assigned to the architectural level of *applications* and the concern *authentication*.

The *Authenticator* pattern [Schumacher et al. 2006] is an abstract security pattern concerning authentication [Fernandez-Buglioni et al. 2014]. The two patterns *Distributed and Centralized Authenticator* can be seen as specializations of it. *Password, Biometric and Card-based Authenticators* are *Centralized Authenticators*, whereas Authenticators based on the *Credential* security pattern, such as *Token or Certificates*, are *Decentralized Authenticators*. [Fernandez-Buglioni et al. 2014] The *Risk-Based Authenticator* pattern is a *Centralized Authenticator* in this context, see figure 1.

The *Security Session* pattern creates a session after a user has authenticated successfully [Schumacher et al. 2006]. Afterward, a session reference is used instead of authenticating a user repeatedly. It can be used to achieve Single Sign-On (SSO), where the session is validated for each request.

In consumer web applications, Token Authenticators are widely used to support SSO among applications. OpenID Connect [Sakimura et al. 2014] is the de facto standard for this kind of distributed authentication. Authentication using a password is used dominantly for centralized authenticators. When performing tasks that demand a high security level, a Two-Step Verification is typically performed [Grosse and Upadhyay 2013]. Thus, an additional authentication factor is used. This factor is often the mobile device of the user. ID Cards, verified email accounts, biometric authentication using the user's mobile phone, QR-Codes, Near Field Communication are evolving authentication methods.

While solutions in the area of consumer web applications evolved greatly in the past years, security patterns have been described rarely. We see the need for a retrospect to understand and evaluate the new solutions. In particular, security patterns concerning the balance between usability and security requirements need to be discussed, because they may help enterprises to build competitive web applications.

3. RISK-BASED AUTHENTICATION

In this section, we discuss several sources that we used to describe the Risk-Based Authenticator pattern. Furthermore, some of the sources discuss advantages and drawbacks of applying the pattern.

Due to the high impact on usability, authentication mechanisms gathering data on the context of a request and evaluating a probability of a hijacked account were introduced in consumer web applications. Grosse and Upadhyay [2013] call these context factors "somewhere you are", "some way you behave" etc. similar to typical authentication factors, i.e. something you know/have/are. Additionally, they introduce their Two-Step-Verification as alternative to One-Time-Passwords in enterprises.

Further, they discuss the topic of usability versus security. Based on the account type, they discuss the need of strong authentication. They present an approach called device centric authorization. Each client device has its own identity. This identity has the right to access the account. Only if a new device is registered a strong authentication is required by using the already registered devices.

In Golan et al. [2005] a system and method for risk-based authentication is defined. They argue that identity theft has reached an epidemic level. Fraudsters are adaptive and modifying their methods quickly to exploit new vulnerabilities. Furthermore, many authentication solutions have an imbalance in usability versus security. They state that users who do not succeed in authentication themselves may abandon the transaction, which can mean to abandon the service itself. Beside this, a user also may call the customer service which results in higher support costs.

Eckersley [2010] studies the uniqueness of a browser, which is closely related to the aspect of using device information to authenticate a user. In the sample of this study 94.2% of browsers with Flash or Java enabled were unique. However, the fingerprint changes quite rapidly. Motivated by this fact, they present an algorithm which is able to guess and follow many fingerprint changes. This algorithm correctly predicts the predecessor fingerprint to a new fingerprint in 99.1% of cases. Therefore, a device is a very good authentication factor of a user.

In Dlamini et al. [2012] scenarios for cloud-based authentication are presented. They describe the risk associated with specific requests, e.g., if a user requests subsequent access to a resource using a registered device, the risk should evaluate to be low. If the user uses a device for the first time, the risk must be adjusted to "medium low". Furthermore, they propose a cloud-based authentication architecture.

3.1 Authentication Indicator

Based on the previous sources, we discuss the term authentication factor in the context of a risk-based authentication and we give a definition for the term authentication indicator.

Authentication factors are the key for authentication methods as proof of authenticity. The traditional view on these factors is user centric and, therefore, most authentication methods can be separated into three types of authentication, by knowledge, possession or characteristic. In ubiquitous computing environments, including web applications, there is always an amount of uncertainty in the user identification [Hulsebosch et al. 2007]. Even when using inherence factors, the uncertainty exists.

This leads to the additional usage of context data to verify the subject's authenticity. "Somewhere you are", "some way you behave", as mentioned in [Grosse and Upadhyay 2013], refers to this kind of information. It can can not be directly associated with traditional authentication factors. Thus, we suggest to use the term authentication indicator in the context of web applications. The term is already used by the MIT Kerberos & Internet trust (MIT-KIT) Consortium [2016] in a similar way for the Kerberos authentication protocol.

Definition 3.1 (*Authentication Indicator*). Any information that may verify the authenticity of a subject. This especially includes context information like the behavior or location of a subject as well as its change in comparison to previous observations.

4. RISK-BASED AUTHENTICATOR PATTERN

4.1 **Summary**

The RISK-BASED AUTHENTICATOR (RBA) pattern uses information on a subject, so called Authentication Indicators, to verify their risk of not being authentic and, in case of a high risk, it takes further steps to raise the trust in the subject's authenticity. A profile is generated for each subject by continuously and transparently gathering information on them during a session. A risk, that the subject is not authentic, is calculated for each request to the RBA by comparing the authentication indicators with the profile of the subject. If the risk exceeds a defined limit a step-up or reauthentication may be scheduled to reduce the risk or the access is denied.

4.2 Example

We want to build an online shop for shoes that addresses consumers with a web application through the internet. Several other shops exist and, thus, usability is one of the key quality characteristics we want to address while still being secure. Users shall offer their personal data as simple as possible and authenticate with easy to use methods. We use a weak authentication based on social networks (social login) because it is easy to use. But, if a risky action is performed, e.g., if the user wants to initiate payment, we want to be sure they are authentic and their account is not hijacked.

4.3 Context

Web applications that offer services for consumers and that are accessible through the internet. The consumer needs to authenticate in order to use the services. Transactions of the web application need to be secured properly.

4.4 Problem

In general, we are aware of vulnerabilities in our authentication, e.g., passwords can be stolen, fingerprints can be imitated or the authentication protocol implementation may have a security flaw. Thus, hijacking user accounts is a problem in our web application. How can we prevent attackers from impersonating as a legitimate service consumer and, as a result, causing damage to the user and our enterprise? The following forces have to be tackled by the solution:

— **Usability:** Ease of use is a competitive advantage and users will switch to a competitor in case of bad usability.
— **Effort:** The user should be authenticated with little effort. If the effort is too high, the user will not use the application.
— **Frequency:** The user should interact as often as needed but not more. The frequency of user interaction to authenticate directly affects the usability and should be kept small.
— **Performance:** The response time of requests in the web application as well as the web user interface's response time should keep almost the same when using a transparent authentication.
— **Cost:** The authentication should be cost-effective, e.g., the need for hardware raises costs and does not scale well.
— **Security level:** The authentication should ensure a high security level, thus, the probability that the user is not authentic is small.
— **Security:** The information used to authenticate the user must be securely stored to avoid unwanted access.

4.5 Solution

You should continuously and transparently collect information on users and learn about their typical context. Based on the gathered information, calculate a risk for a concrete request context. If the risk is high, process a case handling for the request, e.g., schedule a step-up authentication, inform the account owner or deny access. Perform a risk-based authentication for a request based on your requirements, e.g., if a valuable asset is accessed or on each request.

4.5.1 ***Structure.*** In Fig. 2 the structure of the pattern is depicted. The *RbaAuthenticator* realizes an Authenticator. Thus, it authenticates a *User* and, in case of success, it creates a *ProofOfIdentity*. The *RbaMediator* that the user is not authentic, and for giving a recommendation on how to process their request. For this purpose, the *RbaAuthenticator* forwards each request to the *RbaMediator*.

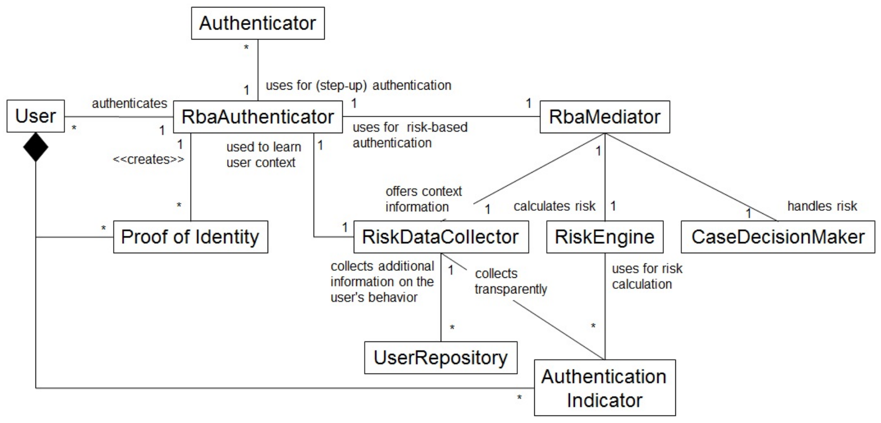

Fig. 2. Structure of the Risk-Based Authenticator based on the Authenticator pattern [Schumacher et al. 2006]

The *RiskDataCollector* collects *AuthenticationIndicators* and the subject's history information from the http request and *UserRepository*. The *RiskEngine* takes the *AuthenticationIndicators* and calculates a risk for the current request. The *CaseDecisionMaker* decides how to handle the request based on the risk score. The *RbaAuthenticator* processes the request taking the recommendation in mind, e.g., it invokes an authenticator or destroys a previously created *ProofOfIdentity*. It may act against the decision of the *CaseDecisionMaker* under certain circumstances, e.g., if it shall reauthenticate the user, although this does not reduce the risk score.

4.5.2 ***Dynamics***. The flow of the RBA is separated into an abstract flow, that handles the authentication without knowing about risk, and the flow mediated by the *RbaMediator*. We separated the flows of the RBA into three use cases. The use case "Intercept a request" describes the abstract flow while the use cases "Register user" and "Risk-based authenticate user" describes the mediated flow.

The first use case "Intercept a request" focuses on handling further steps based on the risk score, Figure 3. The *RbaAuthenticator* processes the http request to the *RbaMediator* which returns a recommendation for handling the risk. Afterward, it handles the recommendation, authenticates the user, takes further steps with contacting other *Authenticators* or rejects the request.

The other use cases describe how the *RbaMediator* handles the first request (use case "Register user") and how a typical request ("Risk-based authenticate user") of a user is processed using the *RiskDataCollector*, *RiskEngine* and *CaseDecisionMaker*, Figure 4.

4.6 Implementation

There are three areas to focus on when implementing the RBA. First, the AuthenticationIndicators, that shall be used to calculate the risk score, have to be determined. Second, all information needed to authenticate the subject based on the AuthenticationIndicators as well as methods to gather this information have to be defined. Third, for each possible risk score the recommendation of the CaseDecisionMaker has to be specified.

The three areas are discussed using a simplified implementation of a campus web application. The application offers services for students. They can see an overview of their grades by authenticating with their matriculation number and a password. The grades our web application offers have a high demand for confidentiality.

1. Authentication Indicators need to be chosen.

We have to choose authentication indicators (hereafter only "indicator") according to the requirements of our web application. The demand for confidentiality is high, thus, we need to choose indicators that assure

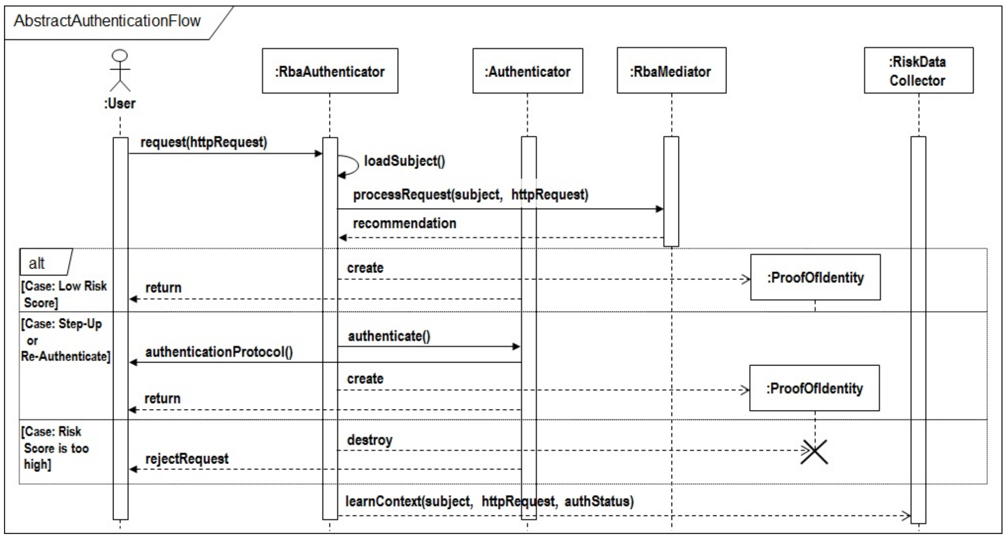

Fig. 3. Sequence diagram for use case 1 “Intercept request and process according to risk score”

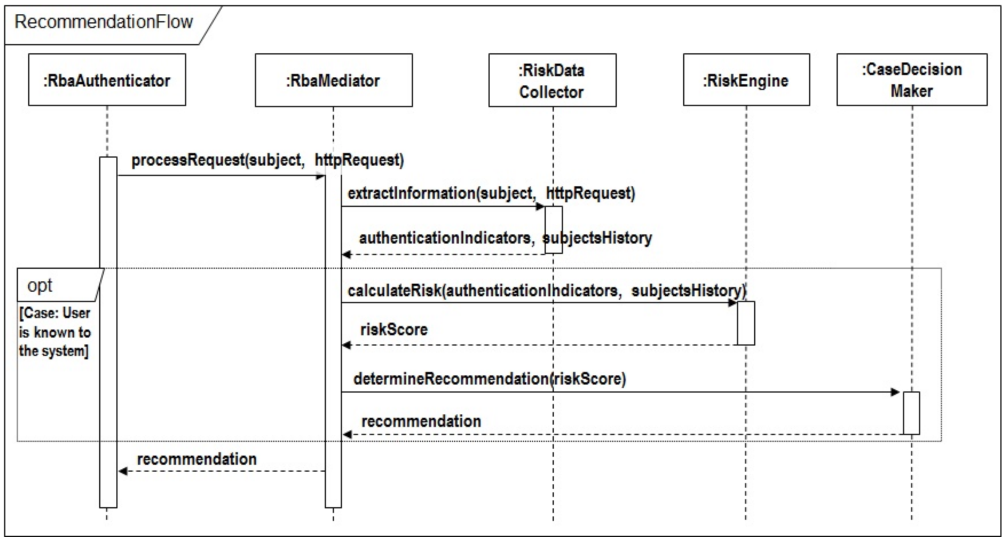
RecommendationFlow
:RbaAuthenticator
:RbaMediator
:RiskData Collector
:RiskEngine
:CaseDecision Maker
processRequest(subject, httpRequest)
extractInformation(subject, httpRequest)
authenticationIndicators, subjectsHistory
opt
[Case: User is known to the system]
calculateRisk(authenticationIndicators, subjectsHistory)
riskScore
determineRecommendation(riskScore)
recommendation
recommendation

Fig. 4. Sequence diagram for use case 2 “Register subject and learn basic context” and use case 3 “Process risk-based authentication request”

a high trust in the user’s authenticity. We decide to take a browser fingerprint, because studies show that these fingerprints identify users with an accuracy above 90% Eckersley [2010]. Additionally, we take the geo location of the request into consideration and count the failed authentication requests. In our simplified scenario, a change of the fingerprint shall result in a high risk score. Requests from foreign countries shall raise the risk score and reduce the count of allowed authentication attempts.

We took the design decision, that the risk score is a ratio scale ranging from 0 to 100 [percent]. According to the previous considerations, we define the following risk calculation, that our *RiskEngine* implements.

The RiskEngine calculates a sub risk score for each authentication indicator and sums these sub risk scores up, while the maximum is 100. - The sub risk score of the browser fingerprint indicator is 100, if the browser fingerprint changes, and otherwise 0. - The sub risk score of the authentication attempts indicator is $20 * |failedAttempts|$. - The sub risk score of the geo location indicator is 60, if the user tries to access from a foreign country, and otherwise 0.

2. The RiskDataCollector must collect the relevant information to calculate the risk.

In our scenario, we have three indicators to collect - browser fingerprint, authentication attempts and geo location.

We use an open source framework (https://github.com/Valve/fingerprintjs2) to calculate a browser fingerprint and add it to each request in the HTTP header. The fingerprint is recorded on the user's first login, where they have to use an initial password they received by encrypted mail. Implementing the authentication attempts indicator is quite simple. The value is initiated to 0 and reset on successful authentication. For each unsuccessful attempt, the value is increased by 1. The geo location is resolved using a free web service (http://ip-api.com/). The IP address of the user's HTTP request is send to this service. The answer of the service includes the country (geo location) of the IP address.

3. The CaseDecisionManager has to be configured to handle the risk score.

Now, we have to define, which risk scores lead to which re- or step-up authentication method and which risk score is too high and results in the denial of the authentication. The risk score has to be harmonized; CaseDecisionMaker and RiskEngine must have the same understanding.

In our example, this is achieved by defining a risk score range. In addition, the risk calculation should be taken under consideration, when defining the risk score handling. As our application has a high demand for confidentiality, we use a strict strategy. Someone accessing our application from a foreign country shall have just one authentication attempt. Requests using an unknown browser shall be declined directly. But, the user can request an initial password at will (send to them encrypted) and add further browsers.

Thus, the CaseDecisionManager is configured to reject requests with an overall risk score higher than 70 and no step-up or re-authentication is scheduled. One failed authentication from a foreign country ($60+20*1 > 70$) and four from the country of our university ($20*4 > 70$) are allowed. Changing the fingerprint always results in a denied request ($100 > 70$).

4.7 Consequences

The RBA pattern offers the following benefits:

— **Effort:** Strong authentication mechanisms often go with a high user effort, e.g., when an extra device generating one-time passwords is needed and must be at hand. The RBA lowers this effort by enforcing a quite strong authentication based on authentication indicators. Additionally, if needed, strong authentication is performed only in case of high risk.
— **Frequency:** The user's authenticity is verified continuously, but transparent to the user. The user does not have to interact with the system.
— **Performance:** The RBA can authenticate the user asynchronously. Thus, an authentication request can be sent to the RbaAuthenticator while the user can go on interacting with the system. This is helpful, when the authentication takes a long time.
— **Cost:** The RBA offers itself a quite strong authentication based on authentication indicators. Thus, it can be used to strengthen weak authentication, such as a password-based, without the need of new hardware.
— **Security level:** By applying the RBA, The Password Anti-Pattern [Crumlish and Malone 2009] (identity delegation by revealing user name and password to a foreign service provider) as well as stolen passwords are detected and can be treated.

— **Separation of Concerns:** The RBA separates concerns, thus, data collection, risk calculation and handling of risk is divided into different components.
— **Extensibility:** Existing Authenticators do not need adaptation. They are treated as resource for the RBA and they are used for step-up or re-authentication.

The pattern has the following potential liabilities:

— **Performance:** If the authentication is performed synchronously, the RBA can lead to performance issues. This is typically the case for the first request of a session. Additionally, some authentication indicators may need to collect a lot of data to statistically identify the subject. This may slow down the web application or raise requirements to the subject's client.
— **Privacy:** Sensitive information on the user is collected. This may be communicated to the user, e.g., depending on the law enforced to the web application.
— **Complexity:** The RBA is an authenticator on top of existing authenticators, thus, additional components are introduced by the RBA making the application more complex.
— **Security:** Securing the information used to authenticate the user, including history data, is not part of the RBA. The data store must be secured properly.
— **Security:** The information used to authenticate the user must be securely stored to avoid unwanted access. Additionally, manipulating this data may lead to account hijacking.
— **Storage:** The history data of subjects raises the need for a data store.

4.8 Known Uses

The RBA is implemented in several IAM products, such as the following:

— ForgeRock OpenAM [ForgeRock Community 2016], an open source solution offering several authentication (single-sign on, adaptive authentication) and access control (web services security, entitlements) features. OpenAM offers a web user interface to configure the RBA. The administrator can define how the risk is calculated using given authentication indicators and they can define for different URLs, which risk score is acceptable and how high risk shall be handled.
— CA Risk Analytics [CA Technologies 2016], a commercial product offering authentication based on risk calculation for web and mobile platforms. The system calculates a fingerprint for devices in order to identify subjects. The risk calculation is based on neural networks, learning throughout the subject's session. The software focuses on e-commerce web applications.
— SafeNet Context-Based Authentication [SafeNet Inc. 2016], offers a RBA authentication based on configurable context information, e.g., ip-address, daytime and device identifiers. Their "Context Engine" calculates a context assurance level, which requires a specified authentication level. Each authentication mechanisms is associated with an authentication level. Depending on this information, a step-up authentication may be initiated.
— PortalGuard [PistolStar Inc. 2016], uses multiple factors to identify the subject. Here, the parameters for allowing access are strictly specified, e.g., access shall only be allowed during office hours. If a subject falls outside this parameters, access is denied. These parameters can be defined by the administrator.

Google, as mentioned in [Grosse and Upadhyay 2013], Facebook, the video game developer Blizzard Entertainment and other companies implement the RBA, too. They reject suspicious requests, force to register new devices or schedule further authentication steps in order to strengthen trust.

4.9 See also

— The RBA is an specialization of the Authenticator pattern [Schumacher et al. 2006] using Authentication Indicators.
— The pattern needs at least one concrete Authenticator, such as Password, Biometric or X.509 [Fernandez-Buglioni et al. 2014], that proofs the identity of an subject. Additionally, these Authenticators can be used by the RBA for step-up authentication.
— When using a Security Session pattern [Schumacher et al. 2006], the RBA can be used to verify the authenticity of the user. For example, a change of the user's location in a limited extend is acceptable, whereas, a complete change of the location indicates a high risk and may invalidate the session.
— An Intercepting Web Agent [Steel and Nagappan 2006] may be used to transparently authenticate requests using the RBA without affecting the web application.
— Access Control taking Risk into consideration, e.g. [Chen and Crampton 2012; Ni et al. 2010], is comparable to the RBA. But, in contrast to the RBA, these access control models (being quite similar to patterns) also consider the accessed resource for risk calculation.
— The Information Obscurity pattern [Schumacher et al. 2006] may be used to secure the stored information.

5. CONCLUSION AND FUTURE WORK

We described the security pattern candidate "risk-based authenticator". The risk-based authenticator offers a solution for the problem of vulnerabilities in non-continuous authentication in web applications.

The risk-based authenticator is a first step in describing the recurring solutions for consumer web applications. For example, step-up authentication is a widely spread solution to avoid a strong authentication at the beginning of a user's session. Building a pattern landscape for authentication in web applications for consumers is a helpful tool for software architects in this area.

The area of consumer web applications and their identity and access management solutions is developing fast. The enterprises offering services for consumers are in a hard competition and have a high demand for usability. Thus, new easy to use and despite that secure solutions are already available (and patented) and still will come up in the future. These solutions need further investigation and may be incorporated into the existing security pattern landscape.

ACKNOWLEDGMENTS

We are grateful to the following people for feedback and suggestions during shepherding, writers workshops and in our research group: Eduardo Fernandez-Buglioni (shepherd), Bogdana Botez, Veli-Pekka Eloranta, Benjamin Hippchen, Poonam Ponde, Tobias Rauter, Lukas Reinfurt, Paulina Silva, Andreas Sinnhofer.

REFERENCES

Christopher Alexander. 1977. *A pattern language: towns, buildings, construction.* Oxford University Press.

CA Technologies. 2016. CA Risk Authentication. (2016). Retrieved November 14, 2016 from http://www.ca.com/us/products/ca-risk-authentication.html

Liang Chen and Jason Crampton. 2012. *Security and Trust Management: 7th International Workshop, STM 2011, Copenhagen, Denmark, June 27-28, 2011, Revised Selected Papers.* Springer Berlin Heidelberg, Berlin, Heidelberg, Chapter Risk-Aware Role-Based Access Control, 140–156.

Christian Crumlish and Erin Malone. 2009. *Designing social interfaces: Principles, patterns, and practices for improving the user experience.* " O'Reilly Media, Inc.".

Nelly Delessy, Eduardo Fernandez-Buglioni, and Maria M Larrondo-Petrie. 2007. A Pattern Language for Identity Management. In *Computing in the Global Information Technology, 2007. ICCGI 2007. International Multi-Conference on.* IEEE, 31–31.

M Dlamini, H Venter, J Eloff, and Y Mitha. 2012. Authentication in the Cloud: A Risk-based Approach. *University of Pretoria* (2012).

Peter Eckersley. 2010. How Unique is Your Web Browser?. In *Proceedings of the 10th International Conference on Privacy Enhancing Technologies (PETS'10)*. Springer-Verlag, 1–18.

Cath Everett. 2016. Are passwords finally dying? *Network Security* 2016, 2 (2016), 10 – 14. DOI:http://dx.doi.org/10.1016/S1353-4858(16)30017-4

Eduardo Fernandez-Buglioni. 2013. *Security patterns in practice: designing secure architectures using software patterns.* John Wiley & Sons.

Eduardo Fernandez-Buglioni, Nobukazu Yoshioka, Hironori Washizaki, and Joseph Yoder. 2014. *Abstract security patterns for requirements specification and analysis of secure systems.* Universidad de la Frontera, 437–450.

ForgeRock Community. 2016. OpenAM Project. (2016). Retrieved November 14, 2016 from http://openam.forgerock.org

Gartner Inc. 2016. IT Glossary Identity and Access Management. (2016). Retrieved November 14, 2016 from http://www.gartner.com/it-glossary/identity-and-access-management-iam

L. Golan, A. Orad, and N. Bennett. 2005. System and method for risk based authentication. (May 2005). US Patent App. 10/938,848.

E. Grosse and M. Upadhyay. 2013. Authentication at Scale. *Security Privacy, IEEE* 11, 1 (Jan 2013), 15–22.

R.J. Hulsebosch, M.S. Bargh, G. Lenzini, P.W.G. Ebben, and S.M. Iacob. 2007. Context Sensitive Adaptive Authentication. In *Smart Sensing and Context*. Lecture Notes in Computer Science, Vol. 4793. Springer Berlin Heidelberg, 93–109.

David L. Jobusch and Arthur E. Oldehoeft. 1989. A survey of password mechanisms: Weaknesses and potential improvements. Part 1. *Computers & Security* 8, 7 (1989), 587 – 604. DOI:http://dx.doi.org/10.1016/0167-4048(89)90051-5

Bill McQuaide. 2003. Identity and Access Management. *Information Systems Control Journal* 4 (2003), 35–38.

MIT Kerberos & Internet trust (MIT-KIT) Consortium. 2016. Authentication indicators - MIT Kerberos Documentation. (2016). Retrieved November 14, 2016 from http://web.mit.edu/kerberos/krb5-devel/doc/admin/auth_indicator.html

Qun Ni, Elisa Bertino, and Jorge Lobo. 2010. Risk-based access control systems built on fuzzy inferences. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security.* ACM, 250–260.

NuDetect Solutions. 2016. Good User Verification with Unparalleled Accuracy. (2016). Retrieved November 14, 2016 from https://nudatasecurity.com/nudetect/

OWASP Foundation. 2014. Top 10 2013-A9-Using Components with Known Vulnerabilities. (2014). Retrieved November 23, 2016 from https://www.owasp.org/index.php/Top_10_2013-A9-Using_Components_with_Known_Vulnerabilities

OWASP Foundation. 2015. Top 10 2013-A2-Broken Authentication and Session Management. (2015). Retrieved November 23, 2016 from https://www.owasp.org/index.php/Top_10_2013-A2-Broken_Authentication_and_Session_Management

G. Paul and J. Irvine. 2016. IEDs on the Road to Fingerprint Authentication: Biometrics have vulnerabilities that PINs and passwords don't. *IEEE Consumer Electronics Magazine* 5, 2 (April 2016), 79–86. DOI:http://dx.doi.org/10.1109/MCE.2016.2521978

PistolStar Inc. 2016. Contextual Authentication. (2016). Retrieved November 14, 2016 from http://www.portalguard.com/contextual-authentication.html

SafeNet Inc. 2016. Context-Based & Step-Up Authentication Solutions. (2016). Retrieved November 14, 2016 from http://www.safenet-inc.com/multi-factor-authentication/context-based-authentication

Natsuhiko Sakimura, J Bradley, M Jones, B de Medeiros, and C Mortimore. 2014. *OpenID Connect Core 1.0.* Standard. The OpenID Foundation.

M. Schumacher, Eduardo Fernandez-Buglioni, D. Hybertson, F. Buschmann, and P. Sommerlad. 2006. *Security Patterns âĂŞ Integrating Security and Systems Engineering.* Wiley Series in Software Design Patterns.

Chritopher Steel and Ray Lai Nagappan, Ramesh. 2006. *Core Security Patterns: Best Practices and Strategies for J2EE", Web Services, and Identity Management.* Pearson Education India.

Michael VanHilst and Eduardo Fernandez-Buglioni. 2007. Reverse engineering to detect security patterns in code. In *In Proceedings of the International Workshop on Software Patterns and Quality.* Information Processing Society of Japan, 25–30.