STUDIA INFORMATICA 2015
Volume 36 Number 2 (120)

Robert SEKULSKI, Marek WODA
Wroclaw University of Technology, Department of Computer Engineering

A NOVEL APPROACH TO USERS' AUTHENTICATION AND AUTHORIZATION

Summary. In this paper, an adaptive method of users' authentication and authorization is proposed. With Bring Your Own Device postulate, working conditions and users' habits have changed and the users connect to company resources with their own devices. This situation poses a severe threat to security, but tightening security rules is not always an option. This brought a need of an adaptive system, which would choose methods adequate to the current threat level. Proposed solution not only minimizes the risk of unauthorized access to company's data, but also simplifies users' authentication process.

Keywords: adaptive method, authentication, authorization

NOWATORSKIE PODEJŚCIE DO UWIERZYTELNIANIA I AUTORYZACJI UŻYTKOWNIKÓW

Streszczenie. W niniejszym artykule została zaproponowana nowatorska – adaptacyjna metoda uwierzytelniania i autoryzacji użytkowników. Zbudowany w myśl tego podejścia adaptacyjny system testowy dobiera metody autoryzacji i uwierzytelniania adekwatne do obecnego poziomu zagrożenia. Proponowane rozwiązanie nie tylko minimalizuje ryzyko nieuprawnionego dostępu do danych firmy, ale tak upraszcza proces uwierzytelniania użytkowników.

Słowa kluczowe: metoda adaptacyjna, autoryzacja, uwierzytelnianie

1. Introduction

Internet recently has become an integral part of our lives. In recent Cisco surveys [1] more than half respondents answer that they cannot live without the Internet. Different people name

different reasons of its popularity. The most common responses include: fast access to information on almost every topic, easiness of data share, facilitation of communication between people regardless of the distance between them and variety of on-line services available mostly for free. An ease of data share poses a serious threat to privacy and possibility of confidential data leakage. This applies especially to a various type of business for which any leak of corporate information may be pernicious. It became obvious that there is a need for protecting access to at least some of resources. Todorov [18] describes three security processes that should be used jointly in order to provide protected access to resources. Over the years, many various implementations of the processes were created. Smith [17] indicates "Today's authentication systems evolved from decades of attacks, many of them successful". Blissful ignorance of users combined with new threats released every day, cause that the number of infected computers rises every year [15, 7, 4]. [16] brings concerns about security of Bring Your Own Device trend. The biggest gripe with BYOD is a degree of control which company has over devices, which are not its assets. BYOD postulate along with people's tendency to not respect basic password security policies, such as not to use the same password for various systems [12] entails risks that a hacker will be able to obtain remote access to the victim's account and later will get access to company's confidential data. As it is stated in [1], companies will have to change their current security systems to the ones based on a user, a role and a device type. In order to do so, understanding who is using a device, where it is being used, and what information is accessed, is vital. Currently there is a growing demand for identity management systems that are usually neither universal nor simple to use. They work in the following manner: the more suspicious (non-typical) are activities of a user, the more detailed authentication process is conducted.

There are a number of approaches to adaptive security control – unfortunately, none of them fulfils a desired level of granularity in terms of usefulness and easiness to implement. *Threat-Adaptive Security Policy* [18] mimics the human approach of placing trust on others based on their actions, and despite its reasonable good results, this approach is not the best solution for all systems. It creates a problem of choosing a set of positive and negative patterns of behaviors used for calculating the trust. Also a creation of user's profile may be a complicated task in systems which allows multiple users' devices as users' patterns of behaviors, may vary, based on them. *Self-Adaptive Authorization Framework* [9] integrates with the existing RBAC/ABAC authorization infrastructure to manage its configuration in an automated way. The preventive action is chosen from the set of actions based on a result that comes from evaluation function (e.g. function minimizing costs). It may happen that best solution will not be selected due to its high cost. This approach is complex to implement. *Dynamic* Authorization Model Based on Security Label and Role [11] minimizes complexity

of roles assignments when users need access to resources on different security levels – it is based on *Web Services* and introduces intricate ontology. Proposed *Policy Resolver* manages user rights, obligations, and may be too difficult to implement for the most organizations. *OpenAM* is all-in-one access management platform for protecting any type of resource across enterprise [10] however, it cannot exist solely on its own. It is worth mentioning that *OpenAM* forked from *OpenSSO*, which after acquisition of SUN by Oracle was discontinued in an unannounced policy change [5].

2. Adaptive authentication and authorization

The goal of this study is to propose an adaptive method of users' authentication and authorization. Adaptive means: "having an ability to change to suit different conditions". In this case, different conditions were: different user's devices, different user's profiles and different user's patterns of behaviors. As it was mentioned in the foreword – working environment and users' habits have recently changed and employees may often use different devices from different locations to connect company's resources. Such a situation is a severe threat to security; however tightening security rules is not always an option. This brings a need of an adaptive system, which would choose methods adequate to the current threat level. Adaptive solution would not only minimize the risk of unauthorized access to company's data, but also simplified users' authentication process. Within many types of currently available devices that a user may use at work, mobile devices are the least secure. Mainly due to their size, what makes them easier to be stolen or lost. Following this line of thinking, desktops are the most secure, next are laptops, then tablet mobile devices and the least secure are mobile phones. This assumption is not entirely correct. Mobile phones may be easier to be stolen or lost, but if user was logged off from the system, even when someone unauthorized obtained physical access to the device, he will not be able to pass even the first step of authentication process. At the same time, desktops and laptops may be targets of malicious software, which steal users' credentials, what would allow an adversary to log into the system. Of course such software exist for mobile phones as well as for desktops, but their number is still very low comparing to threats for desktops and laptops [6, 7]. This shows that device type should not be the only factor deciding which authentication methods should be used. Equally important are typical working hours matching user's behavior profile. If someone works 5 days a week between 8am and 4pm, and when he suddenly tries to log into the system at 6pm – authentication process should be more complex for him. Such a profile should be created for each of user's devices. That would increase profiles' accuracy,

as it should be suspicious when user normally uses only his mobile phone after 5pm and suddenly he tries to log in from a desktop PC. The next case when system should start an intensified authentication is during abnormal user's activity at work, e.g. when user starts to behave suspiciously, not as usually, for example starts to copy all data, what he has never done it before and what is described as intrusive action for a role. The authentication complexity in that case should be dependent of the user's current threat level (the degree of security risk). Of course, the method in that case should be also different from methods used to authenticate the user for this session. Adaptive authentication may be also used to provide better users' experience. Some methods may be cumbersome to pass on some devices, while other providing the same security level, are easier to pass. Considering that, different set of methods may be used depending on the user's device type. That practice would speed up users authentication process and provide him more comfort.

3. Proposed solution

In order to provide adaptive authentication, a solution similar to *OpenAM*'s authentication chain was used. During first request in a new session, user's device is being recognized using device fingerprinting and system matches it to an appropriate user's profile. If no profile for such a device exists, a new one is being created. Based on that match, a user has to pass selected authentication methods from the chain to obtain required security level for the current request. After that, he can perform actions to which he has permission granted within a role and a security level. If he tries to perform action, which is defined in his role, but requires higher security level, then additional authentication has to be performed. The security level can be lowered during user's work in the response to his abnormal behavior or after performing suspicious actions. The abnormal behavior means that user uses the system in a different way than usual. The suspicious actions set is defined for each role. Higher security level can be obtained by passing more methods from the authentication chain. Lowering it is done by the system as a response to user's *suspicious actions* / *abnormal patterns of behaviors*.

3.1. Authentication Chain

Methods used in the authentication chain were selected after analysis of weak and strong sides of the most popular authentication methods. From three types of authentication methods, *proof-by-property* methods were considered unusable mainly because they are too

costly by the need of additional hardware readers (e.g. iris recognition). Methods that do not require such devices have insufficient accuracy (e.g. face recognition) or require large samples of data to be able to recognize the user (e.g. keystroke dynamics). Comparing other categories, *proof-by-possession* methods were considered better option. This decision was motivated by the fact that they are more secure as key space is not that limited like in *proof-by-knowledge* methods and their main drawback which is possibility to be lost or stolen can be reduced by adding additional password protection. The cost of the hardware tokens is smaller than the cost of the biometric readers, however in this case it also disqualifies this method from use. The alternative for this method may be a software token for mobile devices or e-mail passwords. In the final solution e-mail with password reminder were used as the third method in the authentication chain as they provide additional authentication factor, which is a need to have an access to user's e-mail account. The *proof-by-knowledge* methods were split into two categories. First of them were visual passwords among which one method is especially worth mentioning. The *Passpoints* method [1], which requires from user to click in specified order at selected points on the image, is not only easy to pass by user and provides higher key space with 6 clicks than 8 characters password, but can be also used as a reverse authentication [13] when user can upload picture selected by himself. To reduce the threat of shoulder surfing different pictures can be used at random for this method. For the reasons described above this method was chosen as the first method in the authentication chain. Other methods like *Passfaces* [8] and *Deja Vu* [3] which require selecting one face/image from the set during multiple steps or *Draw-a-Secret* which requires drawing on the grid have the same drawbacks like the pass-points method (mainly vulnerability to shoulder surfing) and in the same time provide lower key space than a high resolution image. From the second category – text passwords – classic character password method was chosen to be used in the final solution. That method was selected as easy one by the most respondents in the experiment described in [14]. The holder of the second place in that ranking – PIN password method was not used because it has the same characteristics as the character password method but provides smaller key space. The variation of character passwords – partial passwords – requires typing only random characters of password each time – in theory should increase security by delaying eavesdrop of password by an attacker. In practice, however it is not effective when password is dictionary one and in the same time creates the additional threat to security related to the need of storing passwords in a plain text. Due to this, it is better to set some requirements for chars in passwords and use original character passwords method. From the other text methods, techniques like cognitive and associative passwords were proven cumbersome and insecure and were not used in the final solution.

3.2. User's Profiles

The fundamental assumption for this work is that user uses different devices to access company's resources. His behavior may be different depending on a type of device he uses and some devices are considered a bigger threat to company's security than others. For those reasons user profiles have been introduced. The maximum number of profiles per user can be limited to increase security however in the test application this restriction was not used. Each of the profiles belongs to one class, which defines maximal security level, which can be obtained at this profile. Number of existing classes and restrictions of their members are configurable. That was introduced to provide better granularity of access restrictions. To recognize user's profile the *device fingerprinting* method was used. As was described in section 4.2 even the simplest methods are sufficient to provide identification good enough to be used in company's system.

3.3. Threat Calculating Algorithm

Higher security level can be obtained by passing more methods from the authentication chain. Lowering it is done as a response to user's suspicious actions and abnormal behaviors. The main difference between them is that suspicious actions are defined and rigid while user's profile may change during the time what changes a definition of abnormal patterns of behaviors. The motivation to use this combined approach was described in [18]. Each action/behavior defined on those lists can have different severity. That means for each action different number of points can be subtracted from the user's account. Each security level has threshold of points. When user's current amount of points is lower than required by his current security level then the level is being lowered. The differences in points between specified security levels do not have to be constant to create more sensitive security levels. The abnormal patterns of behaviors and suspicious actions may vary in different applications depending on their structure and purpose. In the test-application developed for the purpose of this study they contain following entries: *Abnormal patterns of behaviors*: different working hours, too many actions in specified time interval, and different proportions in access to different resources types. *Suspicious actions*: number of unsuccessful attempts during last authentication process, exceeded time without any action, attempt to access resources which are forbidden for user's role.

4. Implementation

The main goal of this study was to create a solution, which can be easily configured and extended to best-fit different users' requirements. To empirically test base assumptions and validate the core design a proof-of-concept implementation has been done. This chapter describes briefly this implementation and shows how it can be configured and extended. Separation of different parts of the system (Fig. 1) allows users replacing only a small part of the code in order to modify or enhance any chosen functionality.

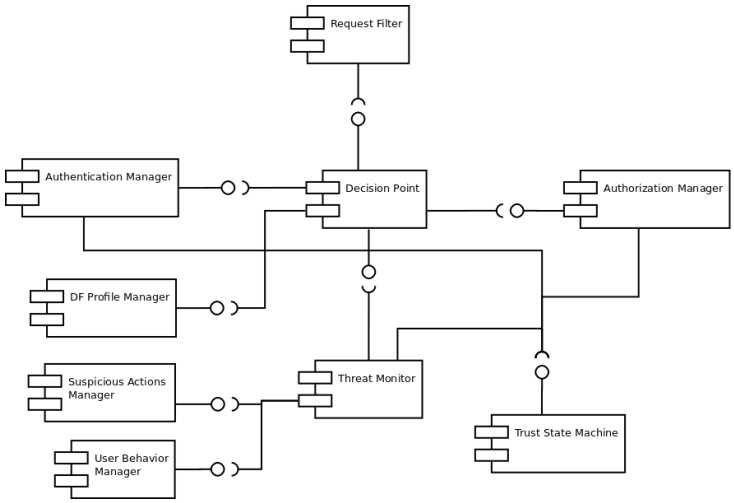

Fig. 1. Main components of the system
Rys. 1. Główne elementy systemu

Request filter

Request Filter (*RF*) role is to intercept all requests to system resources to which access is restricted. It then passes the request to *Decision Point* (section 4.1), which handles it. In the test-application, only one filter was used, however in other situations many filters can be used if filter's role is more complex than just passing requests.

4.1. Decision Point

Decision Point (*DP*) is the core component, which handles whole processing of each request. It dispatches requests to adequate components depending on the current state of the process and processes returned information. To work correctly, system has to load information like device type, role, behavior profile and a list of suspicious actions for current user. The first attempt to recognize the user is using device fingerprinting. When first request in new session is being processed user is being redirected to the site, which tries to collect his device fingerprint. This process is described in section 4.2. When *DP* gets the request with

collected data, it passes it to *DF Profile Manager* (vide 4.3), which returns the best matching profile or information that there is no suitable profile (refer to section 4.7). No matter what was returned from the *DF Profile Manager* in the next step *Decision Point* calls *Authentication Manager* to perform authentication (described in section 4.4). *DP* and *Authentication Manager* use *Authentication Request* and *Authentication Response* to communicate. The first one is being sent when authentication of the user is required and the second one when *DP* passes authentication data obtained from the authentication method. In current implementation they both contain the same data, which is user's profile, *HttpServletRequest* and *HttpServletResponse*, which can be easily extended when additional information would have to be sent in future implementations. Because profiles returned by the *DF Profile Manager* can be mismatched and some of the authentication methods do not require entering a username, just password, while profile has not been confirmed (before first successful login) user has a possibility to inform the system that his profile was wrongly selected. In that case, *Decision Point* will order *Authentication Manager* to authenticate user using default authentication method, which is obligated to require providing username. When user authenticates himself for the first time in new session if his *DF* profile was matched correctly it is being updated. Otherwise, new *DF* profile is being created for the user. Next *Decision Point* makes a decision how to respond to user's request. If user did not pass the authentication then information about unsuccessful attempt is being saved and user has to try again. In order to make decision, *DP* consults Authorization Manager. To communicate they use *Authorization Request* and *Authorization Response*. Similarly, to described earlier *Authentication Request* and *Authentication Response* they carry only data and do not implement any interface. *Authorization Request* contains *HttpServletRequest* and user's profile. It is being sent to *Authorization Manager*, which responds with *Authorization Response* containing *Authorization Decision* and a text message. *Authorization Decision* can be "*ALLOW*", "*DENY*" or "*INSUFFICIENT USL*" and the text message is being used to explain the reason of denying a request. "*INSUFFICIENT USL*" means that user has to pass additional authentication steps to be able to finalize his request. When the request is allowed *DP* calls *Threat Monitor* to check user's action against his behavior profile and the list of suspicious actions (vide section 4.3). It may result in decreasing user's security level what will have impact on user's next requests. After this user's request is being forwarded to its original destination. In case when request was denied, the flow is almost the same but instead of forwarding to the original destination the user is being forwarded to the page with the information that his request could not be finalized. When Authorization Manager decides that the user has insufficient security level then Decision Point sends *Authentication Request* to *Authentication Manager*, which forwards a user to next

authentication step. No interaction with *Threat Monitor* happens in this case as it is considered an internal action of the system.

4.2. Device Fingerprinting

Initially *Device Fingerprinting* method was planned to be used as one of the authentication methods. However, in the final solution *Device Fingerprinting* is used only as recognition to choose correct authentication method for a user. This change was made because as *Device Fingerprinting* method allows generating unique enough identifiers to be used in limited group of users, the system cannot operate only on raw calculated fingerprint. Because data used to generate fingerprint can be changed (e.g. by upgrading a web browser or one of the plugins) system has to allow some errors in matching a profile. Implementation is described in 4.3. This section describes what data and how is collected. After analyzing available papers describing different *Device Fingerprinting* methods, a decision was made to use only basic methods in the example implementation. To collect required data, a user is redirected to the site, which uses *JavaScript* and *Flash* technologies providing required functionalities. The Fingerprint is stored as a hash of the concatenated string values of all collected data.

4.3. DF Profile Manager

Device Fingerprint Profile Manager (*DF Profile Manager*) role is to save, retrieve and update information about user's devices. One of the main system's assumptions is that a user can use many different devices in his daily work. This component manages *DF Profiles*, which are bond with the user's profile. Decision Point communicates with *DF Profile Manager* using an interface which provides methods to create new *DF Profile*, update existing one and get profile which the best matches collected data. The process of collecting data is described in 4.2. The mostly often-used functionality – matching profile based on the data – does not guarantee correct results. The reason is explained in section 4.2. In the example implementation, system first calculates a fingerprint from all collected data and checks if it exists in the database. If yes, then it is the sure match, because fingerprints are marked as unique in the database. In other case, *DF Profile Manager* loads from the database a set of candidate profiles that are the profiles, which have the same hashes of fonts and plugins as the ones, generated from the collected data. For each of the candidate profiles a number of minor points is being calculated. Minor points are being granted for the equal values of the other fields like time zone, agent string, etc. From the set, a profile with the highest number of minor points is being returned. If two or more profiles have the same

amount of minor points then the first one is being returned. In case when the set of candidate profiles was empty, that means there were no profiles with the same hashes of the plugins and fonts strings, a new set is created with the profiles having only one of those strings equals. The procedure of choosing one of the candidate profiles remains the same, however the selected profile has to have at least a specified amount of minor points to be considered a match. In other case, no profile will be returned. *DF Profiles* have classes. A device class is required because it is used to choose correct settings for user's abnormal behaviors and to choose correct authentication chain. These functionalities are described in sections 4.7 and 4.4. Device class is signed to the profile during its creation. In the test-implementation, classification of the profile is done by the *Profile Class Manager*. The requirements for all classes are stored in an XML file. Each class has to have specified name and maximal *USL* (*User Security Level*), which can be obtained by the user using the device from this class. In case when device can be classified to more than one class, the class with the highest possible *USL* is being chosen. Each class can have specified constraints, which should be fulfilled by the device to be classified to the class. Every property can have an accepted value or range of accepted values. To be classified to the class device has to fit in all specified constraints. The configuration should contain one default class, which has no restrictions for those devices, which were not classified to any of the other classes. The last functionality provided by the *DF Profile Manager* – update existing profile – updates all data stored for the profile in the database, but those changes cannot result in changing device's class.

4.4. Authentication Manager

The one of the fundamental parts of this system – adaptive authentication – is realized using an authentication chain composed of the authentication modules. *Authentication Manager*'s role is to manage user's authentication process by selecting appropriate authentication module from the chain specified for his device class. *Authentication Manager* configuration file contains configuration of the authentication chain. Element "modules" contains list of all available modules. The names provided there, are used to load selected module by the Authentication Manager. It is required that one of the modules be marked as a default. Such a module should require from a user to enter a username. Default module is being used as a fallback method when device fingerprinting was not able to match any profile and system has to authenticate an unknown user. Authentication chain is configured for each of the device classes what allows to specify different sequences of methods depending on the device type. This approach allows choosing methods that are more suitable for the device type, its parameters and security. It also allows setting different *User Security*

Levels which user obtains after passing specified authentication step depending on the device type. Use of an *XML* file as a configuration of the available authentication modules creates a possibility to easily add new authentication modules. Convention used in the project assumes that module's class name is the same as the module's name provided in the file and it uses *JSP* with the same name. Each Authentication Module has to implement *Authentication Module* interface, which contains methods *renderForm,* and "*authenticate*", which take *AuthenticationRequest* and *AuthenticationResponse* respectively. The test implementation uses three authentication modules, which are: *character password* – default module, requires from user to provide his username and password, *passpoints* – requires from user to click on the specified points of the image in the specified order. Each user has own image so this method also works as an authentication method of the page to the user, *password sent via email* – sends a randomly generated password to the user's e-mail address, which the user has to copy into the input field. An appropriate module is selected by *Authentication Manager* when *DP* calls its "*performAuthentication*" method. If user's profile has not been loaded then Authentication Manager chooses default module, otherwise it gets user's current USL from *Trust State Machine* (described in 4.6) and selects next module from the chain. The modules are ordered ascending by the *USL*, which they provide. Then it calls module's "*renderForm*" method to start the authentication step. If no module has been found in the chain then Authentication Manager returns that information to Decision Point, which informs the user that his request is impossible. When *Authentication Manager*'s "*checkConstraints*" method is called, it passes the data to the authentication module, which returns the information if method has been passed. In both cases, this information is returned to *DP*, but in case when method has been passed, *Authentication Manager* first updates user's security level in *Trust State Machine*.

4.5. Authorization Manager

AM's responsibility is to decide if a user has a right to undertake an action on specified resources. It makes decisions based on the roles' configuration and the current user's security level. Each user of the system has assigned a role. Each role has configured permissions to specified resources. It is possible to configure required *USL* for specified action on selected resources for each role. When *AM* makes a decision if user's request should be allowed, it first gets user's current security level from *Trust State Machine*. If this level is equal to zero than it means that user's account has been blocked and his request is denied with the message informing about the blocked account. Otherwise *AM* tries to match requested path and method with the permissions provided for the user's role. The permissions are listed in the

same order as they occur in an XML file. If it finds a match then it compares user's security level with the required by the permission and if it is greater, than the decision is "*ALLOW*" else it is "*INSUFFICIENT USL*". If AM did not find any match then request is denied with the appropriate information. Roles configuration is provided to the *AM* by the *Roles Parser*, which loads completely configuration from an XML file at the start of the system. This approach reduces time which would be needed to parse an XML file each time *AM* makes a decision.

4.6. Trust State Machine

USL determines which permissions defined within his role, a user has granted. *USL* is increased after successful completion of the authentication step and it is decreased by *Threat Monitor* (vide 4.7) in some situations. Each of the levels has a number, minimal amount of points, which are required to stay on this level and an initial amount of points, which are granted to the user when he enters the level. *Trust State Machine* holds the information about user's points and his current *Security Level*. It provides methods to set a new *USL* (used by *Authentication Manager*), get the current *USL* (used by *Authorization Manager* and *Authentication Manager*) and to change amount of user's points (used by *Threat Monitor*), which can result in the change of the current *USL*. The configuration of *USL*s is loaded at the system start and provided by *USL Configuration Parser* similarly as it is done in *Authorization Manager*.

4.7. Threat Monitor

Threat Monitor (*TM*) realizes the main concept of this work – adaptive authorization. It monitors all users' requests and reacts accordingly to the provided configuration (vide 3.3) each of the user's requests is checked against his behavior profile and the list of suspicious actions. User has different behavior profiles depending on the class of the used device. After each session user's behavior, profile is updated to better adapt to user's recent habits. Those updates however take into account previous profile, to prevent of rapid changing of a profile what could be a result of a masquerade attack. The other technique used to prevent such attacks is the suspicious actions lists. Those lists are defined for roles and differently than behavior profiles are unmodifiable. Suspicious actions list and user's behavior profile are managed by *Suspicious Actions Manager* and *User Behavior Manager* respectively. *TM* consults them to obtain required information for specified user. Because suspicious actions are defined for roles and are unmodifiable, they are stored in an *XML* file. Each role has actions, which are considered suspicious. Each action has to have a name, a value and

a number of points, which will be deducted from the user's account for this action. It is Suspicious Actions Manager's role to recognize and handle appropriately all suspicious actions configured in the *XML* file. The second component used by *TM* - user behavior profiles - are stored in the database however they are created according to the configuration which is similarly to the suspicious actions defined in the *XML* file. Each *DF* profile class has specified behaviors, which should be monitored by *TM*. Each of the behaviors in the XML file has to have provided name, points and variance attributes. Similarly to the suspicious actions, name of the behavior has to be recognized by *User Behavior Manager*. Points have the same meaning but instead of the value, which is different for each user, configuration of the user's profile allow to define tolerance to the difference between current value and the value stored in the user's profile. Due to the fact, that most of the suspicious actions and user's behaviors are of the different type and should be checked in different situations, *TM*, *Suspicious Actions Manager* and *User Behavior Manager* are strongly coupled. That means that in current implementation design of *TM* is dependent of the suspicious actions and user behaviors, which it should monitor. The main difference between them is that suspicious actions are defined and rigid while user's profile may change during the time what changes a definition of abnormal behaviors. The motivation to use this combined approach was described in [18]. Each action/behavior defined on those lists can have different severity. That means for each action different number of points can be subtracted from user's account. Each security level has threshold of points. When user's current amount of points is lower than required by his current security level then the level is lowered. The differences in points between specified security levels do not have to be constant to create more sensitive security levels. The abnormal behaviors and suspicious actions may vary in different applications depending on their structure and purpose. In the example application developed for the purpose of this study, they contain following entries: different working hours, too many actions in specified time interval, different proportions in access to different resources types, number of unsuccessful attempts during last authentication process, exceeded time without any action, attempt to access resources which are forbidden for user's role.

5. Conducted Tests

Due to the fact that topic of this work is new and specific, most of the available non-commercial solutions are only theoretical works and their implementations are not available. This situation made impossible any type of comparison between the output of this work and solutions described in section 1.1. Because designed system is meant to be highly

configurable depending on the users' requirements, any performance tests have no sense, because they would be valid only for one specific configuration. For those reasons only basic system tests were performed to prove that all elements of the system cooperates correctly. Due to the limited time and no need of providing continuous integration, all tests were performed manually. All tests described in this section were conducted on the system a test system. Most of the described scenarios were tested for all available roles and device classes, List of tested pairs: (role: DEVELOPER, device class: WORK), (role: DEVELOPER, device class: PC), (role: DEVELOPER, device class: MOBILE), (role: ADMINISTRATOR, device class: WORK), (role: ADMINISTRATOR, device class: PC), (role: ADMINISTRATOR, device class: MOBILE), (role: HR, device class: WORK), (role: HR, device class: PC), (role: HR, device class: MOBILE). The only exceptions are second and third tests which were conducted for all roles but only using device from the WORK class. The first conducted test was designed to test stability of the system which means that after passing initial authentication process if user behaves like described in his behavior profile and does not perform any suspicious actions and does not need higher USL then no additional authentication would be required during whole session. For the DEVELOPER and ADMINISTRATOR roles this scenario was realized by passing two authentication steps no matter of the device class and behave according to the prepared behavior profile, accessing only resources from the "/data/" path. For the HR role test looked different depending on the device class. For WORK class second authentication step was passed and GET and POST methods were used on the "/users/" path. For PC class to perform the same actions third authentication step had to be passed and for the MOBILE class only GET actions were performed after passing one authentication method. For all variations of this scenario, system remains stable and no further authentication was required from users. The second test was designed to test system's ability to decrease highly restricted *USL* in a response to minor threat risk. This scenario provides that user will obtain maximal available *USL* and access resources, which require this level. Then he will perform not permitted request and then will try to access the same resources again. Within the configured suspicious actions attempt to access unavailable resources was chosen because unsuccessful login action cannot be performed in this situation and waiting without action for the specified time to pass would violate user's behavior profile and trigger additional decreasing of points. As was mentioned before this test scenario was conducted for all roles but using only a device from WORK class (only for this class, it is possible to obtain the highest *USL*). For ADMINISTRATOR and DEVELOPER roles the test runs were as follows: requested specified resources, passed all three authentication steps, obtained access, requested unavailable resources, requested specified resources, passed third authentication step, obtained access. For the HR role

the flow looks similar with one difference, which was need of passing two authentication steps after requesting unavailable resources, because this action decreased user's security level to 5^{th} and the second step grants 6^{th} *USL*. Presented test run shows that system responded correctly in the tested scenario. The third test scenario was similar to the second one that means it started with the request to the resources requiring the highest *USL* and then user committed a suspicious action, but instead of trying to access the same resources again user accessed other resources available for his role. In addition, this scenario was tested only for the WORK class. The test runs for this case looked similar to the previous test, however for the ADMINISTRATOR and DEVELOPER roles no authentication was required before accessing resources not requiring the highest USL and for the HR role only second authentication step had to be passed to use POST method. In all cases, system's responses were correct according to the configuration. The forth and fifth scenarios tested if system blocks users considered a significant threat. First of them provided that user who obtained *USL* allowing him to do his daily work, at some point starts to behave different than usual and performs suspicious actions. Previous tests showed that if such behavior is occasional or minor, then user need to pass additional authentication and can continue his work. However, if this period lasts longer or the threat is more serious, then the user should be blocked. For all roles and classes, test runs look as follows: requested specified resources, passed required authentication steps, acted according to the behavior profile and did not perform any suspicious actions, flooded system with the requests to available and unavailable resources, get blocked. Above test run shows, that system reacted correctly and blocked the user who became a threat. By flooding system by requests to different resources user started to behave different than it was defined in his behavior profile and by trying to access unavailable resources he performed suspicious actions. Depending on the role and device class user was blocked sooner or later what is correct and results from the configuration. The last scenario tested systems resistance to the unauthorized access attempts when a villain tries to pass authentication process using a brute force method. In this case, it is assumed that attacker obtained an access to the user's device but has problems with passing the first authentication step. For all roles and classes, the test runs look as follows: requested specified resources, multiple unsuccessful login attempts, successful login, are blocked. The system works in a way that it decreases user's points related to the unsuccessful login attempts after user finally passes this step. Depending on the user's role, more or less unsuccessful attempts are needed to block user's account. As the above test run shows, system reacted correctly and blocked the user before he was able to access any resources. Tests described in this section did not cover all possible scenarios, however they proved that all system components cooperate correctly and whole system fulfills all the requirements.

6. Conclusions

Having regard to the new trends and changes of working conditions that require more traveling and mobility and in the same time minimizing a risk related with these requirements, the solution described in this paper was designed. Employment of adaptive authentication and authorization let users to omit a complex authentication process when it is not required but in the same time protects the classified data for being misused even by authorized employees. This approach not only increases users' performance and facilitate their work but also improves company's overall security. The designed system complements currently available solutions, joins them and extends to respond to the new requirements like use of multiple devices by one user. As the test-implementation and performed test scenarios showed, taken approach works and all theoretical assumptions are valid and possible to implement. This of course does not mean that a universal solution has been created, the current implementation still has many aspects that should be improved, nevertheless this paper was meant to show the direction in which new security solutions should head to adapt to new demands.

BIBLIOGRAPHY

1. An essential and strategic solution for service provider Wi-Fi deployments (2014) [Online]: http://www.wi-fi.org/file/wi-fi-certified-passpoint-an-essential-and-strategic-solution-for-service-provider-wi-fi
2. Cisco annual security report. (2015) [Online]: http://www.cisco.com/web/offers/lp/2015-annual-security-report/index.html
3. Deja vu Security Overview (2014) [Online]: http://peachfuzzer.com/pdf/Deja_Overview-DejaVuSecurity-Datasheet-2014.f.pdf
4. Computer Virus facts and stats (2014) [Online]: http://cloudtweaks.com/2014/04/cloud-infographic-computer-virus-facts-stats/
5. Is OpenAM or OAM the better fit for replacing OpenSSO? (2013) [Online]: http://www.ssocircle.com/en/1284/openam-oam-the-better-fit-for-replacing-opensso/
6. Mobile Malware report - A New Look at Old Threats (2014) BLUE COAT SYSTEMS [Online]: https://www2.bluecoat.com/ja
7. Overall Statistics for 2013. Kaspersky Security Bulletin. (2013) [Online]: http://securelist.com/analysis/kaspersky-security-bulletin/58265/kaspersky-security-bulletin-2013-overall-statistics-for-2013/

8. Passfaces Technology – Graphical Password Technology (2014) [Online]: http://www.realuser.com/enterprise/resources/what_is_two_factor_authentication.htm
9. Bailey C., D. Chadwick W., Lemos R.D.: Self-adaptive authorization framework for policy based RBAC/ABAC models, Proceedings of the IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing, ser. DASC '11. Washington, DC, USA: IEEE Computer Society, p. 37-44. (2011) [Online]: http://dx.doi.org/10.1109/DASC.2011.31
10. ForgeRock. Open identity stack: Forging a new future for identity and access management (2014). [Online]: http://www.forgerock.com/media/filer_public/e5/16/e516c8d9-ba86-46fd-bef8-386ebe6da11b/whitepaper_openidentitystack.pdf
11. Gao J., Zhang B., Ren Z.: A dynamic authorization model based on security label and Role, IEEE International Conference on Information Theory and Information Security (ICITIS), p. 650-653 (2010).
12. Gkarafli S., Economides A.: Comparing the proof by knowledge authentication techniques, International Journal of Computer Science and Security (IJCSS), Volume 4, Issue 2, p. 237-255 (2011).
13. Hollestelle G., Schuurmans T.: Online authentication methods. Evaluate the strength of online authentication methods (2008) [Online]: http://staff.science.uva.nl/~delaat/rp/2007-2008/p30/report.pdf
14. Irakleous I., Furnell S. M., Dowland P. S., Papadaki M.: An experimental comparison of secret-based user authentication technologies. Information Management and Computer Security, 10(3): p. 100-108, (July 2002).
15. Mendyk-Krajewska T., Mazur Z.: Problem of network security threats, 3rd Conference on Human System Interactions, p. 436-443 (May 2010).
16. Miller K., Voas J., Hurlburt G., BYOD: Security and privacy considerations, IT Professional, vol. 14, no. 5, p. 53-5, Sept.-Oct. (2012) [Online]: http://dx.doi.org/10.1109/MITP.2012.93
17. Smith R.: Authentication - from passwords to public keys. Addison-Wesley, (2002).
18. Todorov D.: Mechanics of User Identification and Authentication: Fundamentals of Identity Management, 1st ed. AUERBACH, (June 2007).
19. Venkatesan R., Bhattacharya S.: Threat-adaptive security policy, Performance, Computing, and Communications Conference. IPCCC 1997. IEEE International, p. 525-531, (Feb 1997), http://dx.doi.org/10.1109/PCCC.1997.581559

Omówienie

Niniejszy artykuł opisuje jedną z możliwych odpowiedzi na nowe wymagania – system złożony z adaptacyjnych modułów uwierzytelniania i autoryzacji użytkowników. Podobnie jak w tradycyjnych rozwiązaniach korzystających z ról, każdy użytkownik należy do grupy, która ma zdefiniowane uprawnienia do wyznaczonych zasobów. Adaptacyjność systemu polega na tym, że w zależności od aktualnego poziomu zaufania do użytkownika ma on tylko wybrane uprawnienia spośród zdefiniowanych dla jego roli. Poziom zaufania do użytkownika wzrasta, gdy przechodzi on kolejne kroki procesu uwierzytelniania. Jednocześnie, gdy użytkownik zacznie stanowić zagrożenie dla systemu (wykonując akcje uznane za podejrzane lub zachowując się inaczej niż zazwyczaj), poziom zaufania do niego będzie obniżany, co w niektórych sytuacjach może skończyć się zablokowaniem jego konta. Istotnym czynnikiem z punktu widzenia proponowanego systemu jest urządzenie, z którego korzysta użytkownik. Zależnie od niego dobierane są metody uwierzytelniania oraz ustalany jest maksymalny poziom zaufania, który może zostać przyznany. Powyższa funkcjonalność mogła zostać zaimplementowana dzięki wykorzystaniu metody *Device Fingerprinting*, pozwalającej na identyfikację urządzenia, wykorzystując udostępniane przez nie informacje. W artykule została opisana przykładowa implementacja proponowanego systemu. Ze względu na charakter problemu, znalezienie idealnych ustawień dla wszystkich zastosowań nie jest możliwe. Z tego powodu przykładowa implementacja w dużej mierze jest konfigurowalna przy użyciu plików XML oraz jest złożona z małych modułów, które można w łatwy sposób zastąpić. W związku z tym, że praca dotyczy nowego tematu i większość ze znalezionych podobnych prac nie posiadała implementacji, niemożliwe było przeprowadzenie jakichkolwiek testów porównawczych. Na pokazowej implementacji zostały wykonane jedynie testy sprawdzające logikę systemu, które przyniosły spodziewane rezultaty. Ponadto, należy zaznaczyć, iż przedstawiona praca miała na celu jedynie zaprezentowanie możliwego podejścia do tematu nie zaś dostarczenie gotowego rozwiązania.

Addresses

Robert SEKULSKI: Department of Computer Engineering, Wroclaw University of Technology, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland, sekulski89@gmail.com
Marek WODA: Department of Computer Engineering, Wroclaw University of Technology, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland, marek.woda@pwr.edu.pl