Authentication System for Websites with Paid Content: An Overview of Security and Usability Issues

Adam Hurkała, Jarosław Hurkała

Warsaw University of Technology, Institute of Control & Computation Engineering, Poland

Summary
Every company that delivers paid content online is faced with one essential challenge of protecting its content from illegal access and theft. As the number of users increases detecting and stopping intruders who illegally access a website becomes more and more difficult. While many research focus on security issues of online banking, e-shopping and e-government websites, there is very little study of what is affecting websites with paid content. This paper outlines the most common security threats and explains how to design a secure and user-friendly authentication system for websites with paid content.

Key words:
security, authentication, usability, websites, paid content

1. Introduction

Paid content refers to content on the Internet – such as text, images, video and downloads – which is paid for [10]. This typically indicate materials that cannot be accessed without a payment. Paid content can very often be found on websites that charge subscription rates. Those websites usually allow their visitors to view samples of materials, but charge a fee for full version of those materials.

Websites with paid content are especially vulnerable to cyber-attacks. If an attacker gains unauthorized access to a website then file hosting services and Peer-to-peer (P2P) networks may be used as a means to distribute or share files without consent of the copyright owner [1]. In such cases one individual uploads a file to a file hosting service for others to download. In 2004 there were an estimated 70 million people participating in online file sharing [12].

There are several steps that one can take in order to prevent usage of paid content in an unauthorized manner. Some of those methods require preemptive actions such as using Digital rights management (DRM) solutions [2]. Other methods require taking actions when the harm is done e.g. by reporting content that warrants removal from search engines or file hosting services based on applicable laws.

In this paper we focus on hardening the authentication system for reducing its surface of vulnerability. Section 2 describes authentication principles and presents methods that can be used for protecting content on a website. Section 3 presents an overview of security threats and describes methods of mitigating them. In Section 4, we cover the security and usability requirements of authentication system for websites with paid content. Section 5 concludes the article.

2. Authentication

2.1 Basics

Authentication is, in an information security sense, the set of methods that can be used to establish a claim of identity as being true [11]. It is crucial to note that authentication only settles whether “you are who you say you are” and it does not infer or imply about what the person being authenticated is allowed to do.

There are three categories, referred to as authentication factors that can be used to establish one's identity [11]:

1. Something the user knows – this includes passwords, pass phrases, PINs, answers to secret questions, or in particular any information that a person can remember.
2. Something the user has e.g., ID cards, security tokens (hardware or software), email accounts or cell phones.
3. Something the user is or does, which is based on the relatively unique physical attributes of an individual e.g., fingerprint, retinal pattern, signature, face, voice or other biometric identifier.

Multifactor authentication uses one or more of the factors described above e.g. a website may require users to provide a password (knowledge factor) and a secret code which is delivered to the mobile phone (ownership factor).

2.2 Basic Access Authentication

Basic Access Authentication (BA) is a standard method for enforcing access control to protected web resources which is commonly used on websites with paid content. A HTTP client (e.g. a web browser) can provide a username and password using HTTP Authorization header.

The BA mechanism provides no confidentiality protection for the transmitted credentials, which are encoded using base64 algorithm, but not encrypted or hashed in any way.

Manuscript received July 5, 2013
Manuscript revised July 20, 2013

In many cases it is recommended to access web-pages protected with BA only via a secure (HTTPS) connection.
Because HTTP Authorization header has to be sent with each request, the web browser caches the credentials to avoid constant prompting user for the user name and password. On the other hand web-server needs to verify user identity on each request which is not very efficient e.g. loading a html page with 100 images requires handling of at least 100 authentication requests.
It is also crucial to note that Basic Access Authentication does not provide a portable method for a user to "log out" with the exception of closing the web browser.
Because of security, efficiency and usability issues of BA nowadays cookie-based authentication methods are getting more popular among websites with paid content.

2.3 Cookie-based authentication

A web cookie is a small piece of data sent from a website and stored in a user's web browser. Authentication cookies are the most popular method used by web servers to know whether a user is logged in or not, and to tell on which account the user is logged on.
Authentication cookie might store:

1. Session data – which typically might be just a user name.
2. Session identifier – which is usually a long, random string that links to session data stored on the server side.

If cookie stores a session data, it is recommended to encrypt the data using strong cryptography algorithms to prevent data tampering. Additionally session identifiers should be generated using a secure random number generator and be long enough to decrease the probability of obtaining a valid one by means of a brute-force search. In Table 1 we present a comparison of methods of generating a session identifier in 30 popular open-source software. The most important finding one can conclude from this table is that software developers prefer methods that are easy to implement even though some of those methods might be not secure. Some platforms indeed do not provide an easy to use interface for generating cryptographically strong random numbers, but on the other hand there are plenty of open-source and cross-platform libraries such as OpenSSL that can be used for this purpose.
The security of a cookie-based authentication generally depends on how the session data was generated, how it is stored and how it is transmitted. Security vulnerabilities may allow a cookie's data to be read by a hacker, used to gain access to user data, or used to gain access to the website to which the cookie belongs (see section 3.4, 3.6). For websites that require high level of security it is recommended for the server to accept authentication cookie only from original client's IP address and to transmit the cookie only via secure (HTTPs) connection.

3. Security threats

3.1 Common security threats

There are hundreds of issues that could affect the overall security of a website with paid content. Some of them are common to all web applications and some are strictly related to web sites with paid content. While many research focus on security issues of e-shopping, online banking, e-government or typical websites [17], [18], [19] there is very little study of what is affecting websites with paid content.
Our study based on security logs from 100 different websites shows that most common security threats that are affecting websites with paid content are:

1. Brute-force and dictionary attacks (93%)
2. Account sharing (70%)
3. SQL injection attacks (35%)
4. Security misconfiguration (33%)
5. Cross-site scripting (20%)

In following sections we describe in details those threats and we propose the way of mitigating them.

Table 1: Comparison of popular methods of generating a session identifier

Method	Security	Efficiency	Ease of Implementation	Popularity
Hardware random number generator	Very High	High	Low	Very Low
Secure block cipher working in CTR mode	High	High	Medium	Medium
Secure stream cipher	High	High	Medium	Medium
CryptGenRandom() WinAPI	High	High	High	High
Linux /dev/random	High	Medium	High	Medium
rand() / random() API	Low	High	High	High

3.2 Brute-force and dictionary attacks

Brute-force is one of the types of password cracking that involves trying every possible combination of characters that the password could be composed of [8]. Brute-force attacks are usually utilized when it is difficult to take advantage of other weaknesses in an authentication system. The rapid growth of computing power and simple to use automated password cracking software has allowed the activity to be taken up even by unskilled individuals.
In general, we can distinguish two types of brute-force attacks:

1. Local brute-force attack
2. Remote brute-force attack

Local brute-force attacks require an attacker to gain access to the password database. This can very often be possible when an attacker exploits SQL injection vulnerability or other security misconfiguration on a website (see section 3.4 and 3.5).
A standard password database contains the list of users and passwords for authentication as shown on Fig. 1. To improve security passwords are usually not stored in clear-text, but instead in digest form [5].

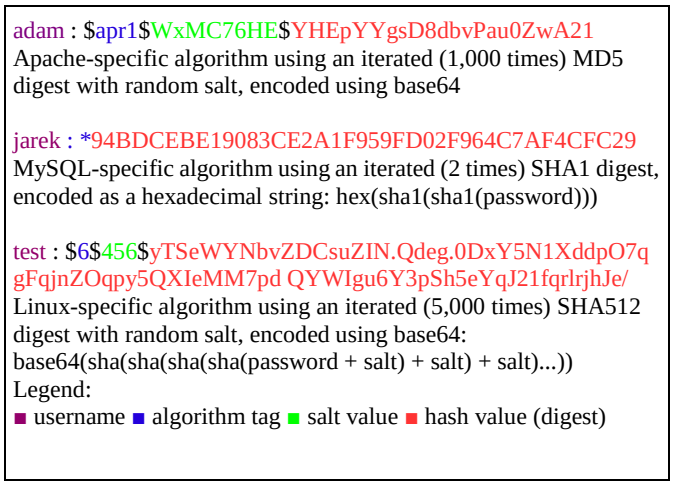

Fig 1. Digests used in popular open-source software

To authenticate a user, the password entered by the user is hashed and compared with the stored hash. To make password cracking even more difficult, the password is often concatenated with a random, non-secret salt value that is stored with the password [19]. Each user might have a different salt, so it is not feasible to compare hashes with precomputed hash values for common passwords.
One of the most popular methods that increases the time required to perform local brute force attack is to use multilevel password hashing (see Fig. 1). Another way of preventing local brute-force attacks is to force a strong password policy, which imposes requirements on what type of password a user can choose. At the time of writing this article random passwords of at least 11 characters containing all printable ASCII characters, hashed using a secure hashing scheme, are not susceptible to local brute-force attacks (see Table 2).
Remote brute-force attacks on the other hand do not require an attacker to get access to the password database. In this type of attack, an attacker is limited by the network latency and usually is not able to check all combinations of login credentials. That is why in a general case, each login-password combination comes from a predefined dictionary that contains frequently used user names and passwords.
A common way to protect a website from remote brute force attacks is to use strong password policy. However, people can no longer remember passwords good enough to reliably defend against dictionary attacks. People also tend to use the same password across multiple sites and once intruders get access to password database on one of those websites they can use the stolen credentials to perform remote brute-force attacks on other websites. Another way of preventing remote brute-force attacks is to limit the number of login attempts per IP address. This method however usually cannot stop attackers who use open proxy servers i.e. servers that act as an intermediary between an attacker and a victim. To successfully prevent automated password guessing a CAPTCHA protection might be used. CAPTCHA, sometimes described as a reverse Turing test, is a type of challenge-response test used in computing as an attempt to ensure that the response is generated by a human [4]. The process usually involves a computer

Table 2: Seconds required to crack a md5 hash on GPU using oclHashcat

	Password Length (in characters)					
Charset	**1-7**	**8**	**9**	**10**	**11**	**12**
lower-case letters (26)	1	38	992	25807	670995	17445878
lower-case letters, digits (36)	14	515	18566	668401	24062468	866248873
lower-case, upper-case letters (52)	191	9773	508209	26426893	1.3742e+09	7.1458e+10
lower-case, upper-case letters, digits (62)	654	39915	2474787	153436812	9.5131e+09	5.8981e+11
all printable ASCII characters (96)	13882	1318810	126605849	1.2154e+10	1.1668e+12	1.1201e+14

asking a user to complete a simple action to prove that the user is indeed a human. These tests are designed to be immensely difficult for a computer to solve, but again easy for a human. If a correct solution is provided by a user, it can be presumed to have been entered by a human. Typical CAPTCHA implementations are distorted text / audio or math puzzles and general knowledge questions [6].

3.3 Account Sharing

Nowadays, user names and passwords for websites with paid content are widely published online on forums and blogs. Sometimes legitimate account owners also share their passwords with friends and acquaintances. Account sharing can cause extreme server load and it can result in a financial loss of a company that delivers paid content. Detecting account sharing can be performed manually by web-server log analysis, but as the number of users increases this task becomes very inconvenient.

Algorithm for automated detection of account sharing should take into account the following issues:

1. Users might share their passwords or session cookies online.
2. User might have a dynamic IP address.
3. User might use several devices such as a computer or a smart-phone to browse the website and each of those devices might have different IP address.
4. User is usually not browsing the website from many various locations (IP addresses) simultaneously.
5. Any HTTP header such as User-Agent or X-Forwarder-For can be spoofed (falsified).
6. Account sharing detection should work in real time.
7. Account should be suspended if and only if there is high probability that it is used by multiple persons.
8. Account sharing detection should not be dependent on any third-party service.

Considering aforementioned assumptions we propose a simple k-sliding-multi-window algorithm for detecting account sharing. In particular, for each user $u \in U$ at time t, we call window w of length p and maximum size s, a list of p pairs, where each pair consist of a unique IP address i and corresponding last access time a, which expires after time e, formally:

$$\mathrm{w}_{\mathrm{k,t,u,p,e}} = \left\{ (\mathrm{i}_1, \mathrm{a}_1), (\mathrm{i}_2, \mathrm{a}_2), ..., (\mathrm{i}_\mathrm{p}, \mathrm{a}_\mathrm{p}) \right\},\ p \le s_k \quad (1)$$

$$(\mathrm{i}_\mathrm{m}, \mathrm{a}_\mathrm{m}) \in \mathrm{w} \wedge (\mathrm{i}_\mathrm{n}, \mathrm{a}_\mathrm{n}) \in \mathrm{w} \Rightarrow \mathrm{i}_\mathrm{m} \neq \mathrm{i}_\mathrm{n} \quad (2)$$

$$(\mathrm{i}_\mathrm{m}, \mathrm{a}_\mathrm{m}) \in \mathrm{w} \Leftrightarrow \begin{cases} a_m + e_k > t \\ \nexists\, (\mathrm{i}_\mathrm{n}, \mathrm{a}_\mathrm{n}) \in \mathrm{w}, \mathrm{i}_\mathrm{n} = i_m \wedge a_n > a_m \end{cases} \quad (3)$$

Account of each user $u \in U$ at time t is represented as a set of k pairs of window w and window size margin α, formally:

$$\mathrm{u} = \left\{ (\mathrm{w}_{1,\mathrm{t,u,p,e}}, \alpha_{1,\mathrm{u}}), ..., (\mathrm{w}_{\mathrm{k,t,u,p,e}}, \alpha_{\mathrm{k,u}}) \right\} \quad (4)$$

$$s_k \ge \alpha_{k,u} > -s_k \quad (5)$$

We defined a boolean function A which for each user determines if the account is shared, based on whether any window size has been exceeded by margin α, formally:

$$A : U \to \{0,1\};\ A(u) = \max_k \left\lfloor \frac{p_{k,u}}{s_{k,u} + \alpha_{k,u}} \right\rfloor \quad (6)$$

We have tested the account sharing algorithm for one month using different number of configuration parameters on one website with 740 user accounts. In order to compare the obtained results with the actual results we have also checked which user accounts are being used by multiple persons by manually analyzing web-server logs.

The results for different number of windows with constant margin α=0 are shown in Table 3.

Table 3: Detecting account sharing with constant margin

Number of windows	Configuration	Correct results	False positives	False negatives
k=1	s_1=2, e_1=10s	51 %	0%	49%
k=2	s_1=2, e_1=10s, s_2=3, e_2=5m	63 %	1%	36%
k=3	s_1=2, e_1=10s, s_2=3, e_2=5m, s_3=4, e_3=1h	70 %	7%	23%
k=4	s_1=2, e_1=10s, s_2=3, e_2=5m, s_3=4, e_3=1h, s_4=6, e_4=24h	83 %	15%	2%

The numbers in Table 3 clearly indicate that using more windows increases overall effectiveness of detecting account sharing, but at the same time increases the number of false positive errors. Because suspending accounts of legitimate users is highly undesirable, we introduced user-specific window size margin parameter $\alpha_{k,u}$ to limit the rate of suspending accounts of users with dynamic IP address. Parameter $\alpha_{k,u}$ was set based on statistics on how often the IP address, within the same Internet Service Provider (ISP), changed for each user. Applying this parameter resulted in better overall effectiveness and less false positive errors as show in Table 4.

Table 4: Detecting account sharing with user-specific margin

Number of windows	Configuration	Correct results	False positives	False negatives
k=1	s_1=2, e_1=10s	51 %	0%	49%
k=2	s_1=2, e_1=10s, s_2=3, e_2=5m	63 %	1%	36%
k=3	s_1=2, e_1=10s, s_2=3, e_2=5m, s_3=4, e_3=1h	71 %	4%	25%
k=4	s_1=2, e_1=10s, s_2=3, e_2=5m, s_3=4, e_3=1h, s_4=6, e_4=24h	89 %	6%	5%

3.4 SQL injection

SQL Injection is a technique often used to attack websites [7]. SQL injection exploits invalid filtering of user supplied data and allows intruders to include a malicious SQL statement that is unexpectedly executed by a database server (see Fig. 2). SQL Injection flaws are introduced when software developers use dynamic database queries that contain user supplied input. Websites with paid content are especially vulnerable to SQL injection attacks because database schema and password hashing algorithm are very often imposed by payment service providers.

A successful SQL injection attack may involve the following security violations:

1. Reading sensitive data from the database e.g. logins, passwords or session cookies.
2. Modifying or deleting sensitive data from database e.g. adding / deleting a user account or session cookies.
3. Accessing system files e.g. accessing software configuration files or user database files.

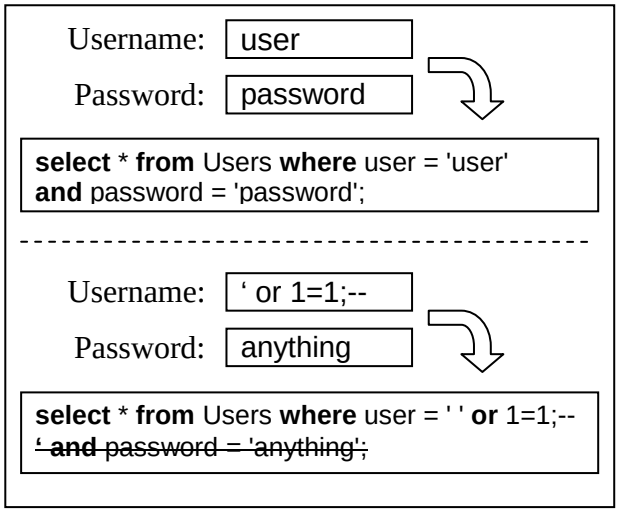

Fig 2. An example of bypassing authentication using SQL injection

In order to prevent SQL injection attacks, one should implement secure coding best practices and reduce or disable debugging information that is displayed on a web-page. Every software developer should also be aware of the following techniques of creating dynamic SQL queries:

1. Prepared statements – software developer has to write a query template with certain constant values that are replaced with user supplied data during each SQL query execution.
2. Stored procedures – software developer writes a procedure that accesses a relational database system in a secure manner i.e. user supplied data is treated as arguments of SQL query.
3. Escaping user supplied input – software developer has to use vendor specific function in order to escape user supplied data e.g. to prepend backslashes to certain special characters.

Other way of securing a website from SQL injection attacks is to use application layer firewall. Typical application layer firewall can provide extended logging capabilities and can monitor the web traffic in real time in order to detect attacks. There are two approaches commonly used by application layer firewalls to prevent SQL injection and other types of network attacks [13]:

1. Negative security model – in this model all requests are monitored for anomalies, unusual behavior and known patterns of web application attacks. Every rule added to the negative security policy increases the efficiency of blocking hacking attempts.
2. Positive security model – in this model only requests that are known to be valid are accepted and everything else is rejected. Every rule added to a positive security model increases what is detected as known behavior, and thus allowed. This approach can usually be applied only for small and medium-sized web applications that are rarely updated and thus is usually not suitable for websites with paid content.

3.5 Security misconfigurations

Security misconfigurations allow an attacker to accesses default system accounts, unprotected files and directories, and to learn important details about a website or the underlying applications.

Common security misconfigurations include:

1. Default administrator password is not changed – attacker can gain full control over a website i.e. attacker can usually add/modify/delete accounts and access log files.
2. Test accounts are not disabled after they are no longer needed – test accounts are an easy target for brute-force attacks because usernames and passwords are very often easy to guess.

3. Directory listing is not disabled on a web-server – attacker can list directories and possibly find sensitive data such as log files, configuration files or user database files.
4. System logs, configuration files, user database or other sensitive data is located in not protected location inside web-server's root directory – attacker might guess filenames and download sensitive files even if directory listing is disabled.
5. Web-application is configured to output various debug information such as stack-trace or database errors, which can be used to learn important details about website and the underlying application or to perform various attacks such as SQL injection.
6. Applications are not removed from server after there are no longer needed – attacker can exploit those applications even though they should have been removed from the server. This especially applies to old applications that are no longer maintained.

Websites with paid content are especially vulnerable, because most of those sites use the same authentication software provided by payment service providers. Our study based on 100 different websites with paid content shows that 70% of websites use the same default configuration without taking any additional security measures.

It is important to note that some of security misconfigurations can be automatically detected by automated scanners and security tools, but there are no tools that can automatically fix those issues. Web server, database and other software behind a website can be protected, however it requires a unified approach from website administrators, software designers and programmers as well as vast knowledge of IT security.

3.6 Cross-site scripting

Cross-Site Scripting (XSS) is a type of attack that enables intruders to inject client-side script into a website and to unexpectedly execute it in the user's web browser [9]. Susceptibility to this type of attack is due to incorrect processing and filtering of user supplied data on web-pages with user-generated content as shown on Figure 3. The task of implementing correct and complete content filter functions might be very difficult, if not impossible. The primary defense mechanism against XSS attacks is to escape untrusted user's input.

Depending on how the output document, which contains user supplied data, is generated, different encoding/escaping schemes must be applied:

1. HTML entity encoding.
2. JavaScript escaping.
3. CSS escaping.
4. URL encoding.

By finding ways of injecting malicious scripts into websites, an intruder can gain elevated access privileges to sensitive page content, session cookies, and a variety of other information stored by the browser on behalf of the user. In order to mitigate the threat of session cookie theft via XSS a HttpOnly attribute might be set for cookie on the server side. If the cookie has this attribute set it cannot be accessed through a client-side script such as Javascript.

Yet another way of mitigating XSS is to accept cookies only from original client's IP address. One should take into account though, that if web-server accepts cookies only from client's IP address some features such as "remember me" might not work properly: e.g. user might be forced to login each time his IP changes.

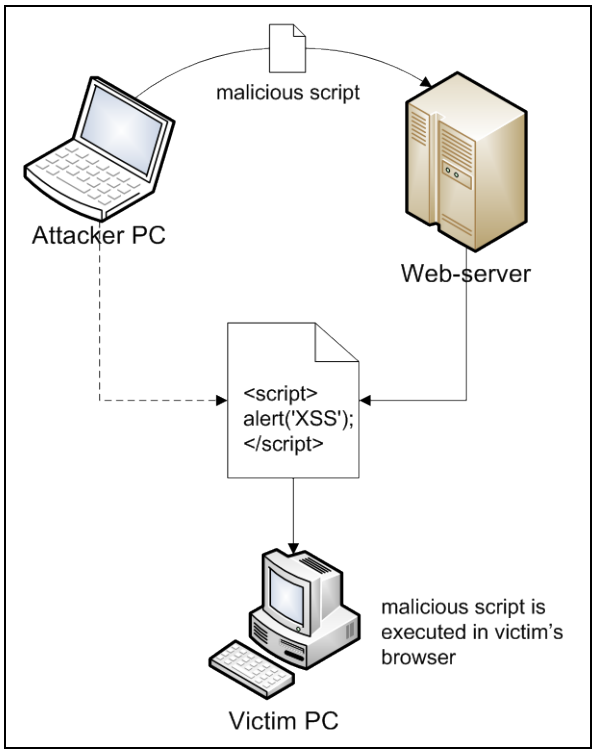

Fig 3. An example of cross-site scripting attack

4. Security and usability requirements

Authentication systems for websites with paid content as any other authentication system have two basic security requirements: they should allow one to access his own account and at the same time they should prevent unauthorized access.

Based on research in web usability [14], [15], [16] we have defined the following five quality components that can be used to evaluate usability of website authentication:

1. Learnability: How easy is it for users to login when they encounter the authentication form for the first time?

2. Efficiency: Once users have learned how to login, how long does it take to authenticate again?
3. Memorability: when users return to the website after a period of not using it, how easily can they regain the desired authentication efficiency?
4. Errors: how many login attempts are typically required for a successful authentication and how easy can a user recover from the authentication errors?
5. Satisfaction: how pleasant is it to use the authentication system?

In order to investigate usability of different authentication methods we tested 6 protection schemes on group of 20 students. Each individual was asked to try and evaluate different open-source authentication software. Quality components were rated on a scale from 1 to 100, where 1 is extremely dissatisfied and 100 is extremely satisfied. The obtained results (see Table 5) were in line with expectations. Meaning of aforementioned quality metric reflect the fact that if an authentication system is too complex, restrictive, or too hard to use, people will not be able or will not want to use it. While majority of people might understand the necessity of confirming a bank transaction using a security token, they most likely will be very upset when being forced to recognize words in highly distorted audio CAPTCHAs when executing an insignificant action. People are used to simple things such as standard passwords and text-based CAPTCHAs and are confused when authentication method changes each time they need to login.

Although CAPTCHA is quite popular solution for ensuring that action is performed by a human, it should also be noted that usability issues of CAPTCHAs were subject of additional researches [3], [4], [6] that showed that they can pose a major accessibility problem to users who are blind or color blind, people with dyslexia, people of advanced age or people with developmental disabilities. On the other hand simple math problems and general knowledge questions are usually also convenient for users, but they present very little level of security unless number of different puzzles is very high.

5. Conclusions

Authentication systems evolved from simple systems, where login and password were transferred using clear-text and stored in unencrypted manner, to complicated solutions which use strong cryptography and secure communication channels. Multifactor authentication, which a few years ago was only used in top-security systems is no longer excess and it slowly becomes a standard of website authentication. CAPTCHA protection, which at first might have seemed unnecessary, can now be found on almost any website with registration form. Any security measures might be inconvenient for website users, but on the other hand they might be necessary. In the end it is the job of IT security architect to know what level of security is required and how to keep the right balance between security and usability.

References

1. Larry E. Daniel, Lars E. Daniel, Chapter 36 - Peer-to-Peer Networks and File Sharing, Digital Forensics for Legal Professionals, Syngress, Boston, 2012, Pages 253-261
2. Pilsik Choi, Sang Hoo Bae, Jongbyung Jun, Digital piracy and firms' strategic interactions: The effects of public copy protection and DRM similarity, Information Economics and Policy, Volume 22, Issue 4, December 2010, Pages 354-364
3. Alessandro Basso, Stefano Sicco, Preventing massive automated access to web resources, Computers & Security, Volume 28, Issues 3–4, May–June 2009, Pages 174-188
4. Ying-Lien Lee, Chih-Hsiang Hsu, Usability study of text-based CAPTCHAs, Displays, Volume 32, Issue 2, April 2011, Pages 81-86
5. I-En Liao, Cheng-Chi Lee, Min-Shiang Hwang, A password authentication scheme over insecure networks, Journal of Computer and System Sciences, Volume 72, Issue 4, June 2006, Pages 727-740
6. Jeff Yan, Ahmad Salah El Ahmad. Usability of CAPTCHAs or usability issues in CAPTCHA design, In Proceedings of the 4th symposium on Usable privacy and security (SOUPS '08). ACM, New York, NY, USA, Pages 44-52
7. Dimitris Mitropoulos, Diomidis Spinellis, SDriver: Location-specific signatures prevent SQL injection attacks, Journal of Computers & Security, Elsevier, 2009, Pages 121-129.
8. Ido Dubrawsky, Jeremy Faircloth, Chapter 2 - General Security Concepts: Attacks, How to Cheat at Securing Your Network, Syngress, Burlington, 2007, Pages 35-64
9. Engin Kirda, Nenad Jovanovic, Christopher Kruegel, Giovanni Vigna, Client-side cross-site scripting protection, Computers & Security, Volume 28, Issue 7, October 2009, Pages 592-604

Table 5: Usability of different authentication methods

	Quality components [1-100]				
Authentication Method	**Learnability**	**Efficiency**	**Memorability**	**Errors**	**Satisfaction**
Username + password	97	99	99	91	98
Username + password + text CAPTCHA	90	92	95	81	86
Username + password + audio CAPTCHA	50	60	71	44	30
Username + password + math puzzle	93	98	98	89	94
Username + password + general knowledge question	89	92	89	88	85
Username + password + one random of above tests	60	65	69	54	30

10. Wojciech Cellary, Paid content a way to electronic knowledge-based economy, in Proceedings of the 14th east European conference on Advances in databases and information systems (ADBIS'10), Springer-Verlag, Berlin, Heidelberg, 2010, Pages 13-14
11. Jason Andress, Chapter 2 - Identification and Authentication, The Basics of Information Security: Understanding the Fundamentals of InfoSec in Theory and Practice, 2011, Syngress, Pages 17-33
12. Ray Delgado, Law professors examine ethical controversies of peer-to-peer file sharing, March 2004, http://news.stanford.edu/news/2004/march17/fileshare-317.html
13. Joseph Migga Kizza, Chapter 11 - Application Proxy, Computer Network Security, 2005. Pages 198-299
14. Sangwon Lee, Richard J. Koubek, The effects of usability and web design attributes on user preference for e-commerce web sites, Computers in Industry, Volume 61, Issue 4, May 2010, Pages 329-341
15. Rafael Tezza, Antonio Cezar Bornia, Dalton Francisco de Andrade, Measuring web usability using item response theory: Principles, features and opportunities, Interacting with Computers, Volume 23, Issue 2, March 2011, Pages 167-17
16. Xiang Fang, Clyde W. Holsapple, An empirical study of web site navigation structures' impacts on web site usability, Decision Support Systems, Volume 43, Issue 2, March 2007, Pages 476-491
17. Jensen J. Zhao, Sherry Y. Zhao, Sherry Y. Zhao, Opportunities and threats: A security assessment of state e-government websites, Government Information Quarterly, Volume 27, Issue 1, January 2010, Pages 49-56
18. Aditya Sood, Richard Enbody, The state of HTTP declarative security in online banking websites, Computer Fraud & Security, Volume 2011, Issue 7, July 2011, Pages 11-16
19. OWASP - The Open Web Application Security Project, https://www.owasp.org

Adam Hurkała received his M.Sc. degree in computer science with honors from the Warsaw University of Technology, Poland, in 2010. Currently he is a Ph.D. student in the Institute of Control and Computation Engineering at the Warsaw University of Technology. His research area focuses on information security, usability, and heuristic algorithms.

Jarosław Hurkała received his M.Sc. degree in computer science with honors from the Warsaw University of Technology, Poland, in 2010. Currently he is a Ph.D. student in the Institute of Control and Computation Engineering at the Warsaw University of Technology. His research area focuses on combinatorial problems, heuristics, fairness and optimization.