Secure Session Management with Cookies

Guy Pujolle[+], Ahmed Serhrouchni[*]
[+]: Lip6, Université Pierre et Marie Curie,
[*]: Institut Telecom, Telecom ParisTech,
Paris, France
Guy.Pujolle@lip6.fr, ahmed@enst.fr

Ines Ayadi,
Lip6, Université Pierre et Marie Curie,
Paris, France
Ines.Ayadi@lip6.fr

***Abstract*— HTTP (Hypertext Transfer Protocol) is a stateless protocol widely used in internet world wide web. The idea behind a stateless design is to simplify the server conception because there is no need to dynamically allocate storage to deal with conversations in progress. If a client dies in mid-transaction, no part of the system needs to be responsible for cleaning the present state of the server. However, this forces web developers to use alternative methods to authenticate HTTP requests and to maintain users' states. A common method for solving this problem involves sending and receiving cookies. Such mechanism implies a serious security threats. Some secure cookie solutions have been proposed in literature, but still vulnerable, particularly to replay attacks. In this paper, we propose a secure cookie mechanism that implements an intermediary reverse Proxy patterns to ensure users' sessions management and to provide the following security services: source authentication, integrity control and no-replay attacks.**

Keyword; HTTP; Cookies; Reverse Proxy; Sessions management; Replay attacks; SSO architecture; HMAC

I. INTRODUCTION

HTTP [1] (Hypertext Transfer Protocol) is responsible for a large volume of today's Internet traffic. Spectacular growth has been earmarked since this protocol has overtaken P2P traffic. Like most network protocols, HTTP is based on request/response transaction model that presumes a reliable transport. Each transaction consists of a client request that invokes a particular method or function on the server and at least one response. HTTP is a stateless protocol that uses external mechanism named cookies to keep track of the state of a transaction. A cookie can be used for authenticating, session tracking (state maintenance), and remembering specific information about users, such as site preferences or the contents of their electronic shopping carts etc. A reason for HTTP success is due to its simplicity and efficiency. However, cookies implementation and HTTP protocol-simplicity make this later vulnerable and easy target for attackers. Cookies do present a serious security threat to information privacy. They can be subject to serious cyber attacks [12] such as session hijacking, cookie poisoning, and cookie replay. Some previous work has been done to address this problem. But these proposed solutions don't implement an application-level mandatory. This paper proposes a secure cookie mechanism that ensures end to end secure connection between user and web servers in network architecture based on Reverse Proxy implementation. This solution provides the following security services: integrity control, source authentication, confidentiality and no-replay of cookies.

The Reverse Proxy [2] patterns implementation seems to be very important to build secure web applications [13]. In the next section, we present the architecture design platform and show its benefits.

II. ARCHITECTURE TOPOLOGY

An intermediary Reverse Proxy implementation that shields the internal web servers is the most widely deployed solution for the following reasons:

- Placing web servers or application servers directly on the Internet gives attackers direct access to vulnerabilities of the underlying platform (application, web server, libraries, operating system).
- A simple packet filter firewall solution still not enough to protect the web servers, since access to its protocol (e.g. port 80 for HTTP) must be provided to the Internet.
- Attack scenarios often use a sophisticated a smart methods, or extra crafted request parameters to manipulate a buffer overflow vulnerabilities.
- Most firewalls perform on the network-application packet level and are not capable to prohibit attacks employing such invalid requests.
- Hardening the web servers can be beyond the administrator's capabilities. Particularly, when the facilities come as a black box from the vendor and call for expertise.
- Switching to another web server software patterns by a different source still an expensive solution, risky and time consuming.
- A new web server might have vulnerabilities that the administrators are not familiar with it.
- Obviously, we are not capable to know about application vulnerabilities and their countermeasures that will be detected in the future.

This proposal architecture implements an intermediary Reverse Proxy that shields the internal web servers. Two packet filter firewalls to ensure that no external network traffic

978-1-4244-4657-5/09/$25.00 ©2009 IEEE

reaches our web servers. Thus, the resulting platform topology provides a demilitarized zone (DMZ) which contains a Reverse Proxy and internal web applications server zone.

As illustrated in the figure 1 the HTTP reverse-proxy is used as an intermediary by Internet clients who want to access internal web servers, by sending it requests indirectly. Thus, all HTTP requests will pass through the reverse-proxy. This platform topology provides robust security policies and implies the following benefits: filtering all HTTP requests, so that only (mostly) harmless requests will reach the internal web servers. Vulnerabilities of the internal web servers can no longer be exploited directly by attackers. Even when the internal web servers get compromised, the firewalls are capable to hinder further spreading of Internet worms by blocking outgoing requests from the internal web servers. Also, even with eventual known vulnerabilities, we might be able to keep the web server configurations stable, since the HTTP Reverse-Proxy with its request filtering policies will be able to prohibit exploitation of those vulnerabilities. Other significant benefit of this architecture is the possibility to address the traceability issue in order to keep track of clients' behaviors and actions.

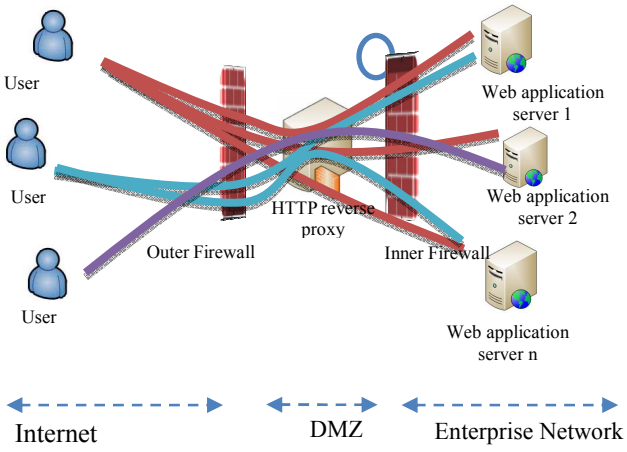

Figure 1. Overall architecture

In this work we handle an HTTP reverse-proxy case. However, the same policies can be applied to any other reverse-proxy application protocol (FTP, IMAP, POP, SMTP, etc.). For instance, FTP protection Reverse Proxy might scan file content for viruses or executable and protect from uploading such files. It can also limit the available FTP commands and prohibit third party host data connections, which are allowed by the ftp standard.

III. HTTP COOKIE: SYNTAX, SEMANTIC AND ANALYSES

HTTP Cookies [3, 4] are a general HTTP state management mechanism. They were initially developed by Netscape in 1994. Netscape's cookies technology led the Internet Engineering Task Force to develop a precise technical standard for cookies (Kristol and Montulli, 2000).

All of the current active cookie specifications implement two complementary HTTP headers, Set-Cookie and Cookie. The Set-Cookie header is sent by the server in response to an HTTP request, which is used to create a cookie on the user's system. The Cookie header is included by the client application with an HTTP request sent to a server, if there is a cookie that has a matching domain and path.

HTTP cookies provide the server with a mechanism to store and retrieve state information on the client application's system. This mechanism provides Web application servers the ability to store information about selected items, user preferences, registration information, and other information that can be retrieved later. Thus, cookies seem to be a strong tool to store all session context and to keep track of users' sessions. However, design and session cookies implementation consist to define a new session state context between a given user and each Web application server. This make the user's session tracking issue a hard task, since a unique user can possess a bunch of HTTP sessions.

In our architecture our concern is to handle a unique session between a given user and a multiple internal application web servers that has access to. In particular we want to address the following forces:

- Different users have different access rights to the internal web servers. We want to be able to handle these differences by a single solution.
- We want to maintain a unique end to end session in SSO architecture based on Reverse Proxy between each user and a multiple internal application web servers.
- We want each Session to be independent of the internal application web servers and their domains.

Set-cookie Header	Cookie Header

set-cookie = "Set-Cookie: ", cookie;
cookie = name, "=", value, {";
",cookie-av};
name = attribute;
cookie-av = ("comment=",value)
| ("domain=", value)
| ("max-age=", value |
"expires=", date)
| ("path=", value)
| ("secure")
| ("version=",{DIGIT}+)
| ("HTTPonly");
	cookie = "Cookie:", [cookie-version],
	cookie-value, {(";" | ","),
	cookie-value};
	cookie-version = "$version", "=",
	value, (";" |
	",");
	cookie-value = name, "=", value,
	[";", path],
	[";", domain];
	name = attribute
	path = "$path", "=", value
	domain = "$domain", "=", value

Figure 2. Set-cookie and cookie headers

The Previous figure presents a grammatical definition of the cookies header in EBNF form [1]. We use the same grammatical definition of the Cookies header field presented in this standard. We will introduce a new attributes that allow handling end to end sessions between users and web application servers in SSO architecture based on Reverse Proxy.

As shown in figure 3, we introduced «session-av» concept by defining a new attributes inside the header cookies. Those attributes are: sessionID, sessionDuration, scomment and ICD.

- "sessionID" it's a field to identify a unique session for a given user. As we mentioned before, the same user might have a multiple activated HTTP sessions. A unique user's session might be defined by associating those valid HTTP sessions in one session. This might be accomplished by introducing a userID parameter or by using other parameters. Such as IP-address, etc.
- "sessionDuration" this attribute is used to define a time period in which a session is valid/accessible.
- "scomment" this attribute is used to add a comments on the current session. This attribute is a mandatory field in our architecture. We will explore the usefulness of this field in more detail in the next section.
- "ICD" (Integrity Cookie Digit) is an attribute used to secure cookies. We will come back in this field attribute in the next section with more details.

RP-Set- cookie Header	RP-cookie Header
set-cookie = "Set-Cookie: ", cookie; cookie = name, "=", value, {"; ",cookie-av, session-av}; name = attribute; cookie-av = ("comment=",value) | ("domain=", value) | ("max-age=", value | "expires=", date) | ("path=", value) | ("secure") | ("version=",{DIGIT}+) | ("HTTPonly"); session-av = ("sessionID=" , value) | ("sessionDuration", value) | ("scomment=" , value) | ("ICD", value);	cookie = "Cookie:", [cookie- version], cookie-value, {(";" | ","), cookie-value}; cookie-version = "$version", "=", value, (";" \| ","); cookie-value = name, "=", value, [";", path], [";", domain]; [";",sessionID]; [";",sessionDuration]; [";",scomment]; [";",scomment]; name = attribute path = "$path", "=", value domain = "$domain", "=", value sessionID = "$sessionID", "=" , value sessionDuration = "$ sessionDuration ", "=" , value scomment = "$ scomment ", "=" , value ICD= "$ICD", "=" , value

Figure 3. RP-set-cookie and RP-cookie headers

Cookies Header will be send to the web servers each time a given user perform a request .This new header contains attributes' values related to user session. Those values allow performing a user session validity and integrity control verification. Our platform topology gives the Reverse Proxy the ability to intercept and process requests send by users before being forwarded to their web servers' destination. Thus, the Reverse Proxy will be able to check whether the user session is valid or not, to verify data-cookies integrity and to prevent from « replay-attack » on cookies.

IV. PROBLEMS AND SOLUTIONS RELATED TO COOKIES

A. Cookies mechanism weakness

Cookies contain information corresponding to a particular context: user, computer, web browser, all domains served by the web server from where it originated. . Besides information privacy concerns, cookies mechanism has some design weaknesses. In particular, they do not always accurately identify users and they pose a real security threats.

1) Inaccurate identification

Cookies mechanism does not identify a user, but a combination of a user account, a computer, and a Web browser. Thus, anyone who uses multiple accounts, computers, or browsers has distinct sets of cookies.

2) Cookie session hijacking

Session hijacking attacks are typically perpetrated in one of two ways: session ID guessing and stolen session ID cookies. Unencrypted cookies can be intercepted by an unauthorized party via packet sniffing. Thus, attacker can hijack the victim's session and compromise the user's session and the sensitive data it holds. This issue can be addressed by employing Transport Layer Security (https protocol) to encrypt the connection. However, in practice, a large number of web servers send session cookies and other data over ordinary unencrypted http connections for performance reasons.

Another different way to steal cookies is cross-site scripting [5] and making the user target itself send cookies to malicious servers that should not receive them.

3) Cookie expiry

The idea behind Persistent cookies is to track and target the interests of users in order to enhance their experience on of services or functions offered sites and to assist them in selected applications. To do this, the web servers retain and correlate information about users between sessions to build up a profile of them over time. This aspect of cookies design also compounds the issue of information privacy, because a stolen persistent cookie can potentially be used to impersonate a user for a considerable period of time

4) Cookie poisoning

Cookie's contents supposed to be stored and sent back to the server unchanged, an attacker may manipulate the cookie's attributes before sending them back to the server in order to bypass security mechanisms of context based trust.. The process of tampering with the value of cookies is called cookie poisoning, and is sometimes used after cookie theft to make an attack persistent. Attackers can gain unauthorized information about a user for purposes of identity theft and other online fraud.

5) Cross-site scripting (XSS)

Cross-site scripting attacks (XSS) [5] often focus on stealing the attributes of the cookie session. This attack is usually performed by forcing the victim browser to run a malicious JavaScript code sometimes including some markup language such as HTML or XHTML. This scripting code is used to transfer sensitive data to the third party. This information

allows the attacker to impersonate the victim or hijack the victim's current session. These attacks are delivered to the victims via e-mail messages or links embedded on other web pages (error pages) or hidden last a picture. When a user clicks on a malicious link, the injected code will be executed by the victim's browser and passed to the vulnerable application server. A different way to exploit stolen cookie is the replay attack. Indeed, knowledge of these cookies can then be exploited by connecting to the same site using the stolen cookies, thus being recognized as the user whose cookies have been stolen.

B. Secure cookies mechanism: state of the art

Many secure cookies mechanisms were proposed to address cookies security issue. We present in this paper four principal secure mechanisms that existed in the literature and their drawbacks related to the security issue:

One-Time Pad Cookie Encryption [6] is a mechanism that manages the distributed storage of sensitive information. For each new cookie or for each update of an old one, a random key will be created. Indeed, when the server generates a new cookie, it generates also a new random key to encrypt the cookie values. Then it put in the database the new encryption key instead the old one associated with the client. When a server receives an encrypted cookie, it find the corresponding key from the database. The major advantage of this solution is the using of a disposable and single use key for each cookie that can limit key attacks. However, this solution has three major drawbacks. First, searching the encryption key, each time the server receives a cookie, implies the overloading. In our architecture, all requests and responses pass by the Reverse Proxy and so this solution can causes the bottleneck effect. Then, if the old encryption key is deleted, all cookies that are encrypted by this key are no longer valid. This represents a major flaw because the user may use a cookie encrypted with an old key before receiving a new cookie and then the request will be rejected by the server. Finally, this solution is mostly used to avoid storing sensitive information. Nevertheless, it seems very important to keep truck of information and actions between the client and the applications' servers.

Other secure cookies mechanism [7] proposes a different way to manage the cookies encryption keys. When the server creates a new cookie, it generates a random encryption key. Then, it encrypts this key with its own public key, and stores the result in the cookie rather than a database. The major drawback of this solution is the decryption of the cookie encryption key each time when the server receives a cookie. This involves a complex and inefficient process.

To provide authentication and integrity of sensitive information stored in cookies. A secure cookies mechanism [8] uses the HMAC function [10]. To do so, HMAC function was applied to the attributes of the cookie and the server secret key. However, this solution doesn't guarantee, neither data confidentiality nor anti replay. Furthermore, using the same secret key represents a source of several types of attacks

To avoid the secret key managing issue, a secure cookies mechanism was proposed in [9]. It uses the output of the HMAC function applied to data cookie as an encryption key. Then, the HMAC function is applied a second time to cookie attributes and the encryption key. This scheme ensures the confidentiality and data cookies integrity. However, applying several times the HMAC function to each cookie might be a time consuming process. Furthermore, this solution does not solve the problem of "replay attacks" because two identical cookies have the same HMAC.

V. PROPOSAL SECURE COOKIES MECHANISM

This section presents a secure cookie mechanism that ensures end to end secure session in network topology based on Reverse Proxy protection. This solution meets the following major security services: source authentication, confidentiality, integrity control, anti-replay attacks protection. This proposal secure cookie mechanism is based on cryptography functions. Basically, the HMAC [10] function will take as input a message M consisting of cookie attributes to protect, an counter value and a secret key that are generated by the Reverse Proxy. Among those cookie attributes we have sessionID, an attribute to identify a user session. Integrating sessionID and the counter value in the message M, is the idea behind this secure cookie mechanism to achieve integrity control, source authentication and anti-replay cookies. The output of the HMAC calculation is then truncated by using HOTP [11] algorithm to obtain an ICD. Because of a traffic optimization and sessions managing reasons, a truncation applied to the output of the HMAC function. To do so, we inspire from the truncation function used in HOTP standard, we obtain a MAC (message authentication code) 6 digits long that we called ICD (Integrity Cookie Digit) through this work.

ICD = truncation HMAC (M||Cpt,Ks)

- truncation HMAC: the truncated output of the HMAC function.
- M= sessionID|sessionDuration|scomment.
- Cpt= a counter that will be generated for each new cookie and incremented at each update.
- Ks: the private key generated and controlled by the Reverse Proxy

Figure 4. The proposal secure cookie mechanism

Input : an RP-cookie

Output: Boolean, set to true if the cookie is valid

1. Decrypt the values of sessionID, sessionDuration, and scomment.
2. Compute the truncation HMAC (sessionID|sessionDuration|scomment||Cpt, Ks), and compare it with the ICD of the cookie. Return true if the two ICDs match.

Figure 5. Cookie verification algorithm

Every time the user sends the secure RP-cookie along with every request. The Reverse Proxy will perform the algorithm (figure 5) to check whether this cookie is valid or not. We precise that both, the counter cpt and the secret key are internal parameters specific to the Reverse Proxy.

VI. SOLUTION DESIGN AND IMPLEMENTATION

In this solution design, the Reverse Proxy implements its own cookie's mechanism to manage users' sessions. A counter CPT is generated by the Reverse Proxy for each user session. We remained that in our topology network the Reverse Proxy is placed between users and the internal web servers.

Let's denote RP-cookie the Reverse Proxy cookies and AS-cookie the server application cookies. Figure below illustrate SA-cookie and RP-cookie headers.

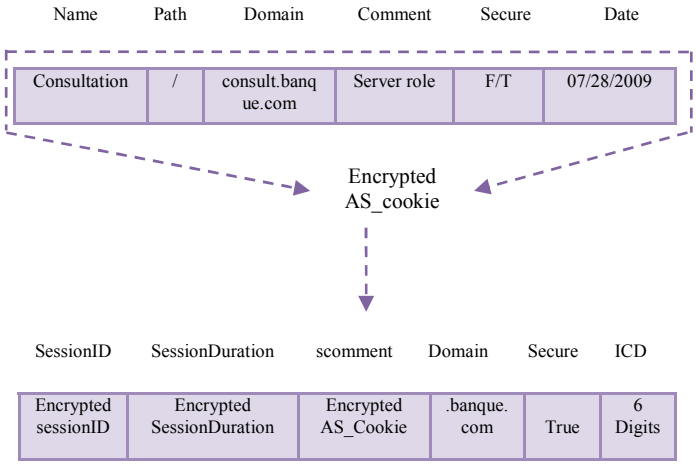

Figure 6. SA-cookie and RP-cookie structures

AS-cookie is encrypted by a secret key and encapsulated inside RP-cookie (figure 6).

Inside the RP-cookie, sessionID and sessionDuration attributes are encrypted by the same secret key. Those attributes are fixed for each single and they are related to session management and the authentication process. This allows achieving a single sign on service.

As "sessionDuration" attribute defines a time period in which a session is valid. It cans be a further protection against replay-attack cookies.

We remark that the counter CPT doesn't exist in the RP-cookie. In fact, the counter value is considered as a session state variable that serves to calculate the ICD. This counter protect against cookies replay-attack since its value is incremented every time the Reverse Proxy generates a new RP-cookie.

RP-cookie has a «secure» attribute that allows a secure HTTP connection based on SSL protocol (HTTPS). One point to note is to set the "Secure" flag when sending cookie over SSL. This ensures confidentiality and the information safety between user and Reverse Proxy.

A typical session between a client and a server consists of two phases: The authentication phase and the Processing HTTP requests Phase

- Authentication phase: In this phase, the Reverse Proxy authenticates the user using the user's login and password, and then generates the CPT counter, the user sessionID and sessionDuration attribute.
- Processing HTTP requests Phase: After user authentication and authorization process. The Reverse Proxy forwards the user request to its web server destination. In response, Set-Cookie is sent by the web server to create a cookie on the user's system to maintain HTTP state session.

Reverse Proxy intercepts a Set-Cookie from the web server, encrypts this cookie by using secret key and encapsulated it in «scomment» field of the RP-cookie. Then, it performs the algorithm (figure 4) to obtain the ICD field value. Every time the user sends the secure RP-cookie along with every request. The Reverse Proxy will perform the algorithm (figure 5) to check whether this cookie is valid or not. We note that for an activate HTTP session a new Set-Cookie might be sent by the web server when the matched received cookie must be updated. The same mechanism will applied on this cookie to perform a user update cookie.

The solution design ensures transparency between users and web application servers. Because implementation does not require any change to the HTTP cookie specification. Figure 7 illustrates our solution implementation.

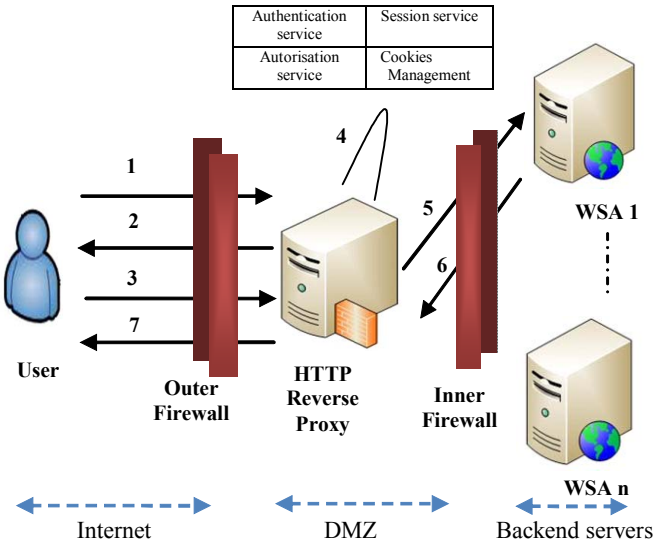

Figure 7. Session's scenario

A. *PROPOSAL SOLUTION BENEFITS:*

- Integrity control: the implemented algorithm functions protect both the cookie integrity as well as its authenticity, by allowing Reverse Proxy to detect any changes to the cookie content.

- Anti-replay protection: because of the CPT counter implementation. The attacker wouldn't be able to replay a valid cookie since this counter is constantly modified and controlled by the Reverse Proxy.

VII. CONCLUSION

Exposing the internal network hosts (applications, web server, libraries, operating system) directly on the Internet presents a serious security treats. Today's network design implements systematically an isolated zone called demilitarized zone DMZ. This zone isolates the internal network with varying levels of protection without compromising its security. A very common secure network design solution places a Reverse Proxy patterns in the DMZ zone that shields the internal network hosts. The existed secure cookie mechanisms don't take advantage of such secure design architecture and still pose the issue of, integrity control, source authentication. Replay-cookie attacks. We have proposed a secure cookie mechanism based on a Reverse Proxy patterns implementation. This solution design ensures transparency, end to end secure session and user session managing. Also, it provides the major security services: source authentication, confidentiality, integrity control and anti-replay attacks protection.

The future work consists to solve the synchronization problem. Related to cookies update mechanism between user and Reverse Proxy.

[1] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners- Lee, "RFC2616: Hypertext Transfer Protocol -- HTTP/1.1,"

[2] Peter Sommerlad, Reverse Proxy Patterns, Procs. of EuroPLoP 2003.

[3] D. Kristol et L. Montulli , "HTTP State Management Mechanism", Internet Engineering Task Force, octobre 2000. [IETF RFC 2965].

[4] David M. Kristol,"HTTP Cookies: Standards, privacy, and politics," ACM Transactions on Internet Technology, vol. 1, pp. 151-198, 2001.

[5] David Endler. The evolution of cross-site scripting attacks. Whitepaper, iDefense Inc., 20. May 2002.

[6] D. Xu, C. Lu, and A. D. Santos. Protecting web usage of credit cards using one-time pad cookie encryption. In Proceedings of the 18th Annual Computer Security Applications Conference, December 2002.

[7] J. S. Park and R. S. Sandhu. Secure cookies on the web. IEEE Internet Computing,2000.

[8] K.Fu, E. Sit, K. Smith, and N.Feamster. Dos and don'ts of client authentication on the web. In Proceedings of the 10th USENIX Security Symposium, August 2001

[9] Alex X. Liu, Jason M. Kovacs Chin-Tser Huang Mohamed G. Gouda, A Secure Cookie Protocol Computer Communications and Networks, 2005. ICCCN 2005

[10] NIST, The Keyed-Hash Message Authentication Code (HMAC), Federal Information Prossessing Standards Publication, FIPS 198a, 6 mars 2002

[11] D. M'Raihi and all.: "HOTP: An HMAC-Based One-Time Password Algorithm", RFC4226, December 2005

[12] David Endler, "Brute-Force Exploitation of web Application Session IDS", IDFENSE Labs, 2001

[13] OWSAP Consortium, www.owasp.org/