SURVEY ON SESSION ANALYSIS

Ms. Nishtha Jatana[*]

Anchal Singhal[**]

Bhavya Sharma[**]

ABSTRACT

The session management mechanism is a fundamental security component in the majority of web applications. It is what enables the application to uniquely identify a given user across a number of different requests and to handle the data that it accumulates about the state of that user's interaction with the application. Where application implements login functionality, session management is of particular importance, because it is what enables the application to persist its assurance of any given user's identity beyond the request in which he supplies his credentials.

Because of the key role played by session management mechanisms, they are a prime target for malicious attacks against the application. If an attacker can break an application's session management, she can effectively bypass its authentication controls and masquerade as other application users without knowing their credentials. If an attacker compromises an administrative user in this way, the attacker can own the entire application.

As with authentication mechanisms, a wide variety of defects can commonly be found in session management functions. In the most vulnerable cases, an attacker simply needs to increment the value of a token issued to him by the application to switch his context to that of a different user. In this situation, the application is wide open for anyone to access all areas. At the other end of the spectrum, an attacker may have to work extremely hard, deciphering several layers of obfuscation and devising a sophisticated automated attack, before finding a chink in the application's armour.

KEYWORDS: Session, security, cookie, session management

[*] Assistant professor, Department of CSE, MSIT, New Delhi, India

[**] Information Technology, Engineering, MSIT, New Delhi, India

I. INTRODUCTION

All over the world Internet users are using web based systems that follows the client server paradigm. The web browser acts as the client for a web server that provides the service. These web based systems rely on the HTTP protocol for communication, but it is a stateless protocol. Therefore, the application developers have to use alternative methods to identify and authenticate the user and maintain the user's state. Both stateless and state full mechanisms can be used with HTTP for session tracking and remembering user specific information. Sessions save the user specific variables and state through consecutive page requests. Sessions are commonly used to enforce security restrictions and to encapsulate run-time state information. A critical issue in web security is the ability to bind user authentication and access control to unique sessions.

Session management allows web based systems to create sessions and maintain user specific session data so that the user does not need to authenticate repeatedly while sending requests. An authentication process is carried out at the first stage to check if the user has the privilege to access the resource. Once authenticated, the user is granted a unique session ID. Thus session management is achieved by maintaining the unique session ID on the client and server side and the browser needs to submit the session ID with every new request.

The existing session management methods are designed originally for a secured environment and the Internet is no longer a secured environment and thus they cannot provide flawless security.

Session management is critical to the security of web based system. Additional measures are needed in the existing session management mechanism to ensure a reliable and sufficient level of session security. One measure is to use strong encryption on all transmissions and to control the session lifetime in a more client way. Special care has to be taken also about generating the session ID to make it unique, unpredictable and hard to regenerate. Handling session data storage and session cookies is another area that needs to be modified to provide better security. Now we are going to explain the underlying concept for session analysis.

LIFECYCLE OF HTTP

A user visits the URL of a website.

This creates a request which is routed to a web server via the internet (a network of DNS's, routers and switches) over HTTP (Hypertext Transfer Protocol). The web server receives the

HTTP request and responds to the user with the web page (or content) which was requested. Every time a link is clicked web page opens, this means you are making a request, and in turn receiving a response from a web server

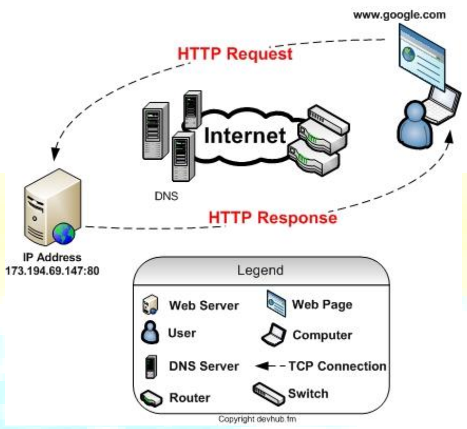

Fig 1: Lifecycle of HTTP

HTTP is a stateless protocol. So, for a plain HTTP request (i.e. without additional information included), a web application is not able to recognize whether this request is related to previous requests or not. It is not able to relate the request to a session for example, what items are in a user's shopping basket or in what stage of a registration process a user is.

The combination of technical mechanisms for solving this task is usually called session management. The most important questions around these mechanisms are:

How to identify the strength of a session?
How secured a session is?

The most popular mechanism is to generate a unique session identifier and store it in a cookie. This identifier can be used for identification as the web browser will send the identifier cookie along with all subsequent HTTP requests to the issuing web application.

HTTP Cookies

Another easy and flexible way of handling session tracking is the use of HTTP cookies. A cookie is a little piece of information that is transmitted from the server to the browser upon session creation. Each time a web client accesses the contents from a particular domain, the client browser will transmit the relevant cookie information along with the HTTP request. In this way, cookies can also be used to store user specific information that can over the user a personalized experience over multiple requests, even multiple sessions. Cookies, containing the expiry information to specify when the browser shall delete it, may last beyond a single session. These types of cookies are stored on the client disk and are called 'persistent' cookies. When a cookie is created without any expiry information, it is only stored in the memory of the client browser and is erased when the browser is closed. These kinds of cookies are known as 'session' cookies.

A stateful HTTP transaction contains two headers for cookies: the SetCookie response header and the Cookie request header. When a client sends a request, the web server includes a Set-Cookie header in the response. This header requests the client browser to include the cookie with all upcoming requests to this web server. If cookies are enabled in the client browser, the browser will add the cookie to all subsequent requests using the header Cookie as long as the cookie is valid. This Cookie header provides information to the server to identify the web client.

A typical cookie-exchange scenario is shown in Figure 1. At first, the web client sends an HTTP request to the web server and the server sends an HTTP response including the Set-Cookie header. After receiving this response, the client includes the Cookie header in the next HTTP request and the server sends back an HTTP response with the requested resource.

A cookie can be set with a name-value pair and some additional attributes.

The name of the cookie is used to identify a cookie for a user and the value is the information, typically the session ID that the web server desires to store in the cookie. Additional cookie attributes are: Domain: The domain attributes species the domain of the web server that created the cookie. This is the same domain for which the cookie is valid.

Path: This attribute is used to refer to the URL of the page on the origin web server.

Max-Age: To specify the lifetime of the cookie, the Max-Age attribute is used by the web server. If this attribute has a value, then the user agent will store the cookie. When the cookie time is expired, the user agent will discard it. Otherwise, the cookie will be deleted once the user closes the user agent.

Secure: This attribute is used to instruct the user agent that the cookie can be sent to the origin server only over a secured connection.

Port: This attribute mentions the port through which the cookie can be returned. Basically, the port tells if the cookie is an HTTP cookie or not.

Session cookies are suitable for storing the session identifiers because the SIDs will be removed from the client when the browser is closed. This type of cookies remain safe even when the user does not log out explicitly and the server is unable to unset the cookie. An advantage of using HTTP cookies over other session tracking solutions is that the server does not need to perform any action to add the SIDs to all the links or forms. The web server generates and transmits a cookie containing the SID to the client and the client browser automatically sends the cookie to the server with each request. In addition, when a server regenerates the SID, the new SID is available to the client browser immediately and there exists no problem with the back or reload buttons. Cookies are also safe to use when the web server is configured not to log the HTTP headers, leaving no traces.

II. PROPOSED WORK

To develop a session analysis tool to help websites analyze their session. Strong session management is a key part of a secure web application. Since HTTP does not directly provide a session abstraction, application and framework developers must make their own using cookies.

The evaluation is mainly based on the following key security requirements for session management mechanisms:

All information that is sufficient for accessing a session must be protected from theft. This information must also be resistant against brute force guessing. Session data must be stored in a way that is safe from manipulation.

III. IMPLEMENTATION

Session analysis is used for analyzing the degree of randomness in security-critical tokens issued by an application. It is typically used to test the quality of an application's session tokens or other items, such as CSRF nonce, on whose unpredictability the application depends for its security.A typical session analysis include:

Send requests that return a security token from other side.

Reissue the same request repeatedly, to generate a large sample of tokens for statistical analysis.

Perform a rigorous set of tests, including the standard FIPS tests and others, to estimate the degree of randomness within the sample, at both the character and bit level.

Start performing the analysis with as few as 100 tokens, and re-perform this as a larger sample is collected, up to the FIPS-recommended sample size of 20,000 tokens.

View an intuitive, at-a-glance summary of all the tests performed, letting you quickly understand the overall quality of randomness.

Review detailed, graphical test output, letting you drill down into the detailed reasons why individual parts of the token passed or failed each test.

Load an existing sample of tokens for analysis, if these have already been captured elsewhere.

It is often highly useful in providing rigorous analysis of an application's session tokens, in cases where these can appear random to both the naked eye and to simpler, scatter-graph based, analyses. It also enables consultants to provide their clients with output to demonstrate that some meaningful work has been done in this often overlooked area of security.

CHARACTER LEVEL ANALYSIS

The Character-level analysis shows the summary result from all character-level tests and lets you drill down into the detail of each character-level test. It also contains charts showing the size of the character set at each position, and the maximum number of bits of entropy that can be contributed from each character position.

- **CHARACTER COUNT ANALYSIS**

This test analyses the distribution of characters used at each position within the token. If the sample is randomly generated, the distribution of characters employed is likely to be approximately uniform. At each position, the test computes the probability of the observed distribution arising if the tokens are random.

- **CHARACTER TRANSITION ANALYSIS**:

This test analyses the transitions between successive tokens in the sample. If the sample is randomly generated, a character appearing at a given position is equally likely to be followed in the next token by any one of the characters that is used at that position. At each position, the test computes the probability of the observed transitions arising if the tokens are random.

BIT LEVEL ANALYSIS

The Bit-level analysis shows the summary results from all bit-level tests and lets you drill down into the detail of each bit-level test. This can let you gain a deeper understanding of the properties of the sample, to identify the causes of any anomalies, and to assess the possibilities for token prediction.

- **FIPS MONOBIT TEST:**

This test analyses the distribution of ones and zeroes at each bit position. If the sample is randomly generated, the number of ones and zeroes is likely to be approximately equal. At each position, the test computes the probability of the observed distribution arising if the tokens are random. For each of the FIPS tests carried out, in addition to reporting the probability of the observed data occurring, Burp Sequencer also records whether each bit passed or failed the FIPS test. Note that the FIPS pass criteria are recalibrated within Burp Sequencer to work with arbitrary sample sizes, however the formal specification for the FIPS tests assumes a sample of precisely 20,000 tokens. Hence, if you wish to obtain results that are strictly compliant with the FIPS specification, you should ensure that you use a sample of 20,000 tokens.

- **FIPS POKER TEST:**

This test divides the bit sequence at each position into consecutive, non-overlapping groups of four, and derives a four-bit number from each group. It then counts the number of occurrences of each of the 16 possible numbers, and performs a chi-square calculation to evaluate this distribution. If the sample is randomly generated, the distribution of four-bit numbers is likely to be approximately uniform. At each position, the test computes the probability of the observed distribution arising if the tokens are random.

- **FIPS RUNS TEST:**

This test divides the bit sequence at each position into runs of consecutive bits which have the same value. It then counts the number of runs with a length of 1, 2, 3, 4, 5, and 6 and above. If the sample is randomly generated, the number of runs with each of these lengths is likely to be within a range determined by the size of the sample set. At each position, the test computes the probability of the observed runs occurring if the tokens are random.

- **FIPS LONG RUNS TEST:**

This test measures the longest run of bits with the same value at each bit position. If the sample is randomly generated, the longest run is likely to be within a range determined by the size of the sample set. At each position, the test computes the probability of the observed longest run arising if the tokens are random. Note that the FIPS specification for this test only records, a fail if the longest run of bits is overly long. However, an overly short longest run of bits also indicates that the sample is not random. Therefore some bits may record a significance level that is below the FIPS pass level even though they do not strictly fail the FIPS test.

- **SPECTRAL TEST:**

This test performs a sophisticated analysis of the bit sequence at each position, and is capable of identifying evidence of non-randomness in some samples which pass the other statistical tests. The test works through the bit sequence and treats each series of consecutive numbers as coordinates in a multi-dimensional space. It plots a point in this space at each location determined by these co-ordinates. If the sample is randomly generated, the distribution of points within this space is likely to be approximately uniform; the appearance of clusters within the space indicates that the data is likely to be non-random. At each position, the test computes the probability of the observed distribution occurring if the tokens are random. The test is repeated for multiple sizes of number (between 1 and 8 bits) and for multiple numbers of dimensions (between 2 and 6).

- **CORELLATION TEST:**

Each of the other bit-level tests operates on individual bit positions within the sampled tokens, and so the amount of randomness at each bit position is calculated in isolation. Performing only this type of test would prevent any meaningful assessment of the amount of randomness in the token as a whole: a sample of tokens containing the same bit value at each position may appear to contain more entropy than a sample of shorter tokens containing different values at each position. Hence, it is necessary to test for any statistically significant relationships between the values at different bit positions within the tokens. If the sample is randomly generated, a value at a given bit position is equally likely to be accompanied by a one or a zero at any other bit position. At each position, this test computes the probability of the relationships observed with bits at other positions arising if the tokens are random. To prevent arbitrary results, when a degree of correlation is observed between two bits, the test adjusts the significance level of the bit whose significance level is lower based on all of the other bit-level tests.

- **COMPRESSION TEST:**

This test does not use the statistical approach employed by the other tests, but rather provides a simple intuitive indication of the amount of entropy at each bit position. The test attempts to compress the bit sequence at each position using standard ZLIB compression. The results indicate the proportional reduction in the size of the bit sequence when it was compressed. A higher degree of compression indicates that the data is less likely to be randomly generated.

IV. Conclusion

The session management mechanism provides a rich source of potential vulnerabilities for you to target when formulating your attack against an application.
Because of its fundamental role in enabling the application to identify the same user across multiple requests, a broken session management function usually provides the keys to the kingdom. Jumping into other users' sessions is good.

Hijacking an administrator's session is even better; typically this enables you to compromise the entire application.

You can expect to encounter a wide range of defects in real-world session management functionality. When bespoke mechanisms are employed, the possible weaknesses and avenues of attack may appear to be endless. The most important lesson to draw from this topic is to be patient and determined. Quite a few session management mechanisms that appear to be robust on first inspection can be found wanting when analyzed closely. Deciphering the method an application uses to generate its sequence of seemingly random tokens may take time and ingenuity. But given the reward, this is usually an investment well worth making.

V. References

[1] The Web Application Hacker's Handbook (2nd Edition) - Dafydd Stuttard and Marcus Pinto

[2] Secure Session Management: Preventing Security Voids in Web Applications- Luke Murphey [January 2005]

[3] Web Application Session Management- created by Secologic Consortium [January 2007]

[4] Session management-Juan M. Gimeno, Josep M. Rib´o [January, 2008]

[5] Improving the Security of Session Management in Web Applications-Philippe De Ryck, Lieven Desmet, Frank Piessens, Wouter Joosen [March 2009]

[6] Phpwn:Attack on PHP Sessions and Random Numbers- Samy Kamkar [August,2009]

[7] Secure Session Management- Fariha Nazmul [June2011]

[8]FIPS-140 random test- Ananda vithanage and takakuni Shimizu [October 2003]

[9]Session Management in Web Applications- EUROSEC GmbH Chiffriertechnik & Sicherheit [May 2005]

[10]Ollmann, G. Web based session management - best practices in managing HTTP-based client sessions. [June 2011].

[11] Secure session management with cookies- Pujolle, G Serhrouchni, A.; Ayadi,I.[December2009]