Redmine - Defect #40648

Session Strength Medium Vulnerability

2024-04-30 20:54 - James H

Status:	Needs feedback	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:	Security	**Estimated time:**	0.00 hour
Target version:			
Resolution:		**Affected version:**	5.1.2

Description

SessionStrength (1)

References
OWASP2021-A02 CWE-330 OWASP2017-A2 OWASP2023API-0xa2

Description
Session tokens that exhibit low entropy ("randomness") and high predictability are often susceptible to prediction attacks. These factors may occur if the
token generation scheme produces values based on: Static data, Incremental data, Time-based values, an Insufficient pseudo-random number generator,
User attributes (e.g. username or some identifier) or other highly deterministic information.
This means that an attacker would be able to guess a valid session token after monitoring the application for a short period of time and gathering the
session tokens it creates. If the attacker determines a valid session token for another user, then it may be possible to view, modify, or delete arbitrary users'
data without having to guess the victim's username or password. Consequently, the ability to deduce valid session tokens enables the attacker to bypass
login pages and obviate the need to brute force accounts.
Additionally, static tokens can enable the attacker to target users even if the victim is not currently logged into the application. This increases the pool of
victims which the attacker can target.
Session tokens should be created with a strong random number generator and gathered from a large pool of numbers. For example, an operating system's
rand() function can usually be sufficient if it can produce 32-bit values that are a statistically uniform distribution. Poor session tokens are incremental, rely
on the user's account ID, only use time stamps, or have other highly deterministic information. Other methods of protecting a session token's security are to
always transmit them over SSL, automatically expire the token after a certain period of time, and explicitly expiring the token whenever a user logs out of the
application.

Recommendation
If the session values exhibit strong randomness, but are chosen from a small pool of values, then the attacker has a better chance of simply guessing a
valid token. A web application's session management can be improved by implementing several complementary techniques:
Make sure that the Token values are at least 32 bits in size, especially for applications with large numbers of concurrent users and high amounts of daily
page requests.
The bit size of the source of the entropy (random values) is more important than the bit size of the actual session token. For example, an MD5 hash
produces a 128 bit value. However, the MD5 hash of incremental values, a timestamp, or 8-bit random numbers are each insecure because the source of the
random values can be easily predicted. Consequently, the 128 bit size does not represent an accurate measure of the session token. The minimum size of
the entropy source is 32 bits, although larger pools (48 or 64 bits) may be necessary for sites with over 10,000 concurrent users per hour.
In most cases, session token names used by the most common web application development frameworks are easy to identify (e.g. ASP.NET_SessionId,
ASPSESSIONID, JSPSESSIONID, PHPSESSIONID) and can disclose the technologies and programming languages used by the web application. It is
recommended to change the default name of the framework to a generic name, such as id.

Track user attributes associated with the session token with server-side objects to prevent user impersonation attacks. If the application does not strictly
associate a user's session token with that user's profile information, then an attacker may be able to view arbitrary information by manipulating client-side
values. For example, if the application sets a strong session token, but performs SQL queries based on a "UserId" cookie, then an attacker only needs to
modify the "UserId" cookie to impersonate someone else. The application would be more secure if it associated the "UserId" value with the server-side
session object because the attacker would not be able to modify the value.
Expire session tokens when the user logs out of the application or after a predetermined period of inactivity. We recommend using a 2-5 minute timeout for
high risk applications, whereas for lower-risk applications, 15-20 minutes would be more appropriate. Session expiration timeout values largely depend on
the type of application and the expected usage.

History

#1 - 2024-04-30 20:55 - James H

low priority

#2 - 2024-05-03 16:57 - Pavel Rosický

Session tokens are generated by https://ruby-doc.org/stdlib-2.7.0/libdoc/securerandom/rdoc/SecureRandom.html which is suitable for this purpose and it doesn't exhibit low entropy issues like those (wrong) variants described in the document. The standard length is 40bit, which follows your 32bit+ recommendation, and session expiration timeouts are configurable.

could you elaborate on why do you think it's a Redmine issue?

#3 - 2024-05-03 18:07 - Marius BĂLTEANU

- Status changed from New to Needs feedback