Initial Post

Password resetting is an expected feature for any website that provides some form of user authentication. When the feature is used, websites will typically send a password reset link via email if a user wishes to change their password (this URL may appear in a format similar to a query URL, e.g. http://example-domain.com/reset_password?token=6fta040f9j). After loading this URL and entering/confirming the new password, the user's password will be changed. Although the process is simple and various solutions exist for implementing this behavior, it is extremely vulnerable to architectural shortcomings which could allow an attacker to gain unauthorized access to user accounts. OWASP classifies these shortcomings as "Broken Authentication" (i.e. category A2), and authentication weaknesses related to password recovery are further described by CWE-640 (MITRE, 2021).

As an example of how this vulnerability can be created, consider the generation of the token for a password reset. Tokens serve as evidence that a password reset request is valid, and should be unique, single-use values. Random Number Generators (RNGs) can be used to generate tokens, however, the RNG in use needs to be properly constructed. Many modern RNGs are pseudo-random, which means that they contain an internal state which is used to generate sequences of seemingly random numbers (NIST, 2010). Due to the reliance on internal state, if that state is replicated, it is possible to begin predicting values that will be generated. In addition, the exact algorithm used to generate numbers can also act as an attack vector- for example, Java's standard library for generating random numbers (java.util.Random) makes use of an algorithm which is easily cracked if six sequential values are found (Oracle, 2020; Haldir, 2004). As a result, if a poor RNG is used, an attacker could figure out the generator's state and begin computing password reset tokens which could then be used to gain control of user accounts. An activity diagram is presented below to demonstrate how an attacker could gain control of user accounts by doing this.

To prevent this, apart from selecting a more secure RNG (more commonly known as a cryptographically secure RNG), it is generally recommended that a RNG's internal state should be dictated by a truly random process (NIST, 2010), and should be reset regularly. Furthermore, limits should be placed on how many password reset requests can be placed in a specific period of time, for a specific user, or alternatively, implement the token generation system according to the microservice architecture, with each microservice's RNG instance having a unique internal state. This is necessary in order to prevent an attacker from gaining a large set of observable values which would act as an attack vector.

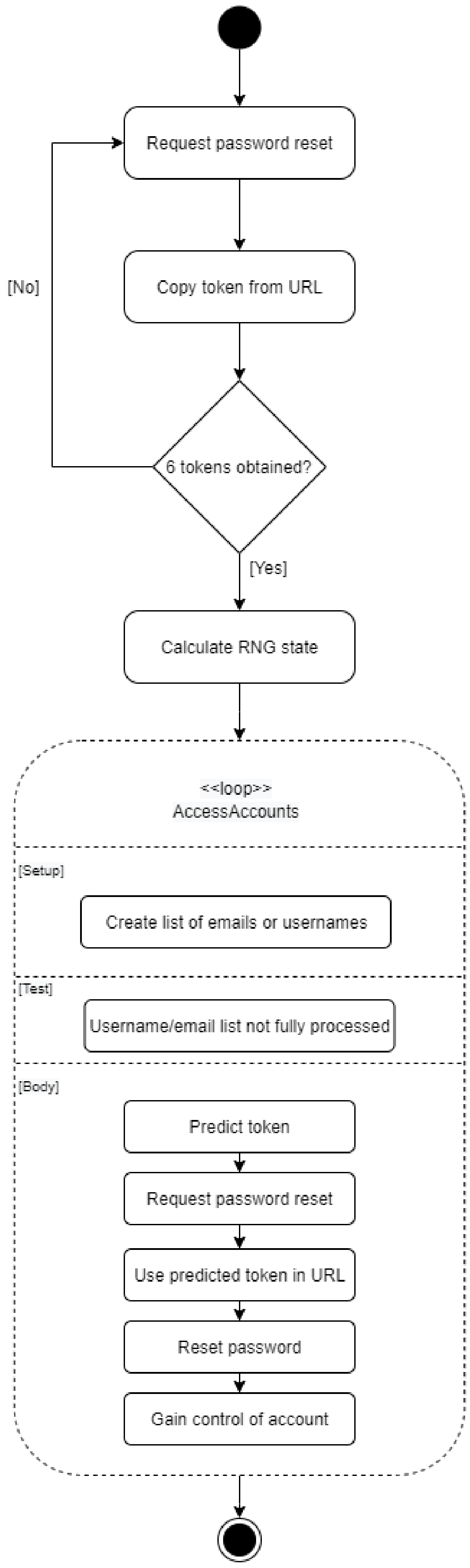
References

Haldir. (2004) How to crack a Linear Congruential Generator. Available from: http://www.reteam.org/papers/e59.pdf [Accessed 18 August 2021].

MITRE. (2021) CWE-640: Weak Password Recovery Mechanism for Forgotten Password. Available from: https://cwe.mitre.org/data/definitions/640.html [Accessed 18 August 2021].

NIST. (2010).A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. Available from: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-22r1a.pdf [Accessed 18 August 2021].

Oracle. (2020) Random (Java Platform SE 8). Available from: https://docs.oracle.com/javase/8/docs/api/java/util/Random.html [Accessed 18 August 2021].


Request password reset
Copy token from URL
[No]
6 tokens obtained?
[Yes]
Calculate RNG state
<<loop>>
AccessAccounts
[Setup]
Create list of emails or usernames
[Test]
Username/email list not fully processed
[Body]
Predict token
Request password reset
Use predicted token in URL
Reset password
Gain control of account