CS4516: Empirical Results

Tag-Based IP Spoofing Prevention Under Low Deployment Scenarios

Author:
Michael Calder
Daniel Robertson

Supervisor:
Dr. Craig Shue

February 27, 2014

1 Results and evaluation of our two approaches

After implementing the hash-chain and TOTP-inspired protocols using NS-3, analysis of the results has shown that both methods offer significant security benefits without causing problems with performance. Each of the two approaches has its own advantages; in this section we will examine these differences and explain how each of the hash functions affect security and performance.

The time interval we used for both methods was about half a minute (32 seconds) but could easily be adjusted to add security or space-efficiency. We chose this time interval because we were never able to crack any collisions in less than half a minute when attempting to brute force a hash. With the hash-chain approach, a compromised hash can only be exploited within the duration of the time interval it was created with. Also, no matter how many hashes we captured, we were not able to determine the original tag (mostly due to collisions) which prevents an attacker from determining future hashes. The only downside of this approach is space complexity. If the interval is roughly half a minute and one day of hashes are generated at a time, 2880 hashes (11KB for xxHash, 46KB for SHA-1) must be stored in RAM. These amounts are not beyond the capabilities of popular routers, but obviously requiring one fourth the space is preferable. Note that as the time interval gets smaller, these numbers grow significantly. Using a 128-bit hash like SHA-1 will also use up four times as much memory as a 32-bit hash like xxHash. With that said, these issues can be negated by increasing how often the hashes are generated (i.e. twice a day).

For the TOTP-inspired approach, each compromised hash also proved useful for a maximum of 32 seconds and then became useless. Because the original tag cannot easily be determined due to collisions (even if the method for combining the timestamp with the tag is known), this method also made stolen hashes unhelpful to attackers in predicting future hashes. Almost no additional memory is needed for this approach and the time interval can be adjusted as desired. The downside to this approach is that, with a lot of work, the original tag can be determined. Using rainbow tables, knowledge of how the timestamp is combined with the tag, and a lot of CPU, having multiple compromised hashes and timestamps can allow an attacker to determine the original tag by comparing the brute-force results until there is only one common tag in the lists of collisions. With that said, breaking

this system now becomes very difficult and time-intensive to acquire only one tag.

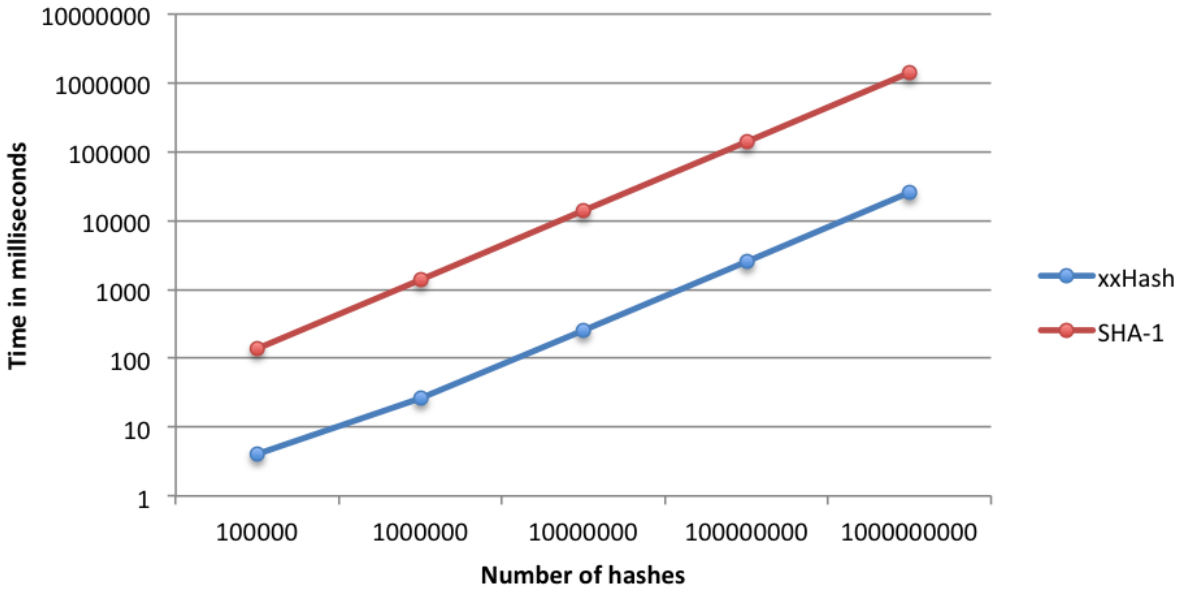

Figure 1: Hash speeds for xxHash and SHA-1 on a 2.4GHz processor

The main differences between cryptographic and non-cryptographic hash functions seemed to be the speed. xxHash, being the fastest non-cryptographic hash algorithm, ran up to 40 million hashes per second. SHA-1, being one of the fastest cryptographic hash algorithms, maxed out around 728 thousand hashes per second. In practice, xxHash caused no measurable latency in the test network (same total time in milliseconds for tag creation/authentication in each router) while SHA-1 failed to meet this requirement. For the once-a-day calculations required for hash chaining, the speeds (on a 2.4GHz processor, which was used to simulate the network) are shown in Figure 1. As far as security is concerned, rainbow tables render both algorithms just as difficult to brute-force. When the original tag is 48 bits, almost 300 trillion possibilities need to be iterated through before a list of all possible original tags that could produce a given hash is known.

So why not use xxHash in any context? xxHash is not a cryptographic hashing algorithm because it is not second preimage resistant, which means that it is possible to determine potential collisions without using brute-force methods. This can be insecure in some contexts, but for what we are trying to accomplish it does not cause an issue because collisions cannot be leveraged. If anything, more

collisions only complicate the process of attempting to reverse each hash in a hash chain or trying to determine the original tag from a TOTP-inspired hash. For the purposes of making compromised tag-hashes not a security concern, we have concluded that both of our approaches (using xxHash) are effective and have the potential to provide any reasonable desired space and time efficiency.