File Verification – Hashes and Signatures

Author	josh
Date	2022-05-20

Overview

Software security not only involves writing good software, but *transporting* software to end users in a secure way. How the final product gets to the end user can expose the consumers of that product to attacks such as man-in-the-middle. Thankfully, there are several tools that reduce risk for those that use software. In this article, we will discuss two common tools for verifying software – hashes and digital signatures, and why both are important.

Validation Tools & Types

Types of Verification We Need

As we discuss the advantages of hash functions and digital signatures, we should discuss more details of what verification actually means. What does "verifying" the authenticity of distributed bits actually mean? For our purposes, there are 2 types we need to be concerned with – *integrity* and *authentication.*

Integrity – Hash Functions
Integrity means, simply, verifying that a message (in this case our software) has not been tampered with in transit. So, for example, ensuring that the software downloaded correctly from a website or passed along from a repository without tampering.

Hash functions are commonly used for this purpose. A website or repository that shares software downloads will often include the *hash* of that binary alongside the software itself. So – the user downloads the software. To verify the *integrity* of that package, they run a hashing algorithm on the file contents. Often times, SHA-256 is used as it is a common, secure algorithm. The user verifies that the SHA-256 hash of the downloaded software matches exactly with the one provided from the website.

Cryptographic hashes are excellent for this as even one bit of difference between files results in radically different values, meaning it's easy to visually and computationally verify the integrity of the software. The hash is also sometimes called the *fingerprint*.

Authentication – Digital Signatures
Using hash functions for integrity sounds like enough, right? Verify the hash/fingerprint means the software hasn't been tampered with from the website, so that must mean it is the legitimate bits we want. Well, not quite. This method does, indeed, verify that the software hasn't been tampered with from website to user.

But what if the website itself is compromised? Maybe an attacker gained access to a web server and replaced both the *software* and the *hash.* How can we be sure our software came from the *intended author?* This is *authentication* – verifying that software came from the expected sender.

Digital signatures are used for this purpose. Digital signatures involve asymmetric cryptography. The software publisher uses their private key to sign the software. Anyone with a copy of the public key can verify the signature, but nobody can create a signed software package without the secret private key. This provides our authentication scheme – anyone can easily verify that a signed package only came from the person with the right private key.

Subtle Differences
The subtle difference here is that it's much harder for an attacker to forge a digital signature than it is to compromise the source of the package *and* hash. If an attacker can gain access to a software download website to replace the file with something malicious, they can also likely replace the verification hash listed on that site. This attack on an integrity check could be slightly mitigated by providing software fingerprints in multiple places such as social media accounts.

It is harder to forge a digital signature, since only the owner of the private key can create a signature against the software package. Signing keys are usually (one would hope) kept somewhere much more secure than a web server. The public key, used for verification, can be shared via PGP servers, websites, and social media. Between multiple locations and trusted parties verifying signing keys, a "web of trust" can be established.

Hashing, Signing, Integrity, and Authentication

Integrity and Authentication are important parts of digital security. When downloading software and packages, one can verify both hashes and digital signatures to ensure the bits are intact and come from the expected source. This not only extends to end-user software, but to libraries used by other software packages. It's important to use these practices to protect users from malicious code.