Software Update Protections Aren't Based on Chance

You've probably flipped a coin to settle a dispute, decide who starts a game, or what side of the field to play on. But when it comes to updating software, it's not a heads-or-tails choice. Software updates must be fully protected on both sides. Software vendors must protect their internal software builds and update processes. And when updates are distributed to customers, they must be immediately installed on their applications, servers and endpoints to mitigate possible breach exposure.

The consequences of lax update protocols

Most of us recall the Equifax data breach that exposed the Social Security numbers, birth dates, and home addresses of 143 million Americans. Hackers accessed the credit reporting agency's data through a known web application vulnerability. Two months prior to the breach, a fix for this security hole was available, but the company failed to update its software. Consequently, the company must now pay at least $575 million, and potentially up to $700 million. This is part of a settlement with the Federal Trade Commission, the Consumer Financial Protection Bureau, and 50 U.S. states and territories.

While updating software to protect clients and servers seems like a simple solution to protecting vulnerabilities, there are obstacles. For various reasons, system administrators can neglect or delay updates. Sometimes they simply can't account for every computer using their network. They may also skip patches out of concern that the new update might break the current version's features. Additionally, they might have shadow IT, with no visibility to know about, or receive the updates.

A trust layer of protection for resources with security vulnerabilities

To protect resources that aren't updated, zero trust network access (ZTNA) clearly defines access control policies for business-critical IT resources, applications, data, and services. Zero trust has a flexible authentication model that controls which user population has access to certain resources. This zero trust cybersecurity model eliminates implicit trust and replaces it with explicit, real-time adaptive trust levels for just-in-time, just enough access to digital resources. The "explicit trust zone" is between the policy decision and enforcement point for applications, servers, systems, services and data.

RevBits ZTN provides another layer of protection that helps eliminate risks associated with zero day and software security flaws. Updates are a continuing security problem for organizations, and RevBits ZTN helps protect apps,

The "explicit trust zone" is between the policy decision and enforcement point for applications, servers, systems, services and data.

servers and endpoints during vulnerable software update periods, with zero trust access to all digital resources under management. Users are limited to what they can do on the resource, and resource IP addresses are concealed, and never exposed. Bad actors will not even be able to discover an asset via network scan if they attempt to identify active devices.

Granular access to specific corporate resources is enabled without exposing the entire network. This significantly limits the potential for malicious actions. With critical resources, either on-premise or in the cloud, and access demands coming from all directions, RevBits ZTN empowers companies with the necessary access security, regardless of user, device, or access location.

RevBits ZTN includes natively integrated privileged access management (PAM), with identity-based authentication, multi-factor authentication (MFA), single sign-on (SSO), end-to-end encryption, session recording and more. Remote access authentication and authorization protect resources inside the network, and encrypted tunnels secure connections for outside network traffic.

How secure are your vendor's software builds and internal update processes?

It's important for organizations to know if the software they receive from a vendor is secured and protected. Software vendors must have safe internal product builds and update processes to prevent hackers from infiltrating their products and injecting malware into updates that will be distributed to customers.

A few years ago, the North Korean government obtained access to a third-party South Korean cybersecurity provider and inserted malware into their updates and stole American military secrets. It's believed that someone may have been paid or threatened, and left the Internet cable connection in place. This example shows how software updates can be manipulated by insiders and those outside the organization.

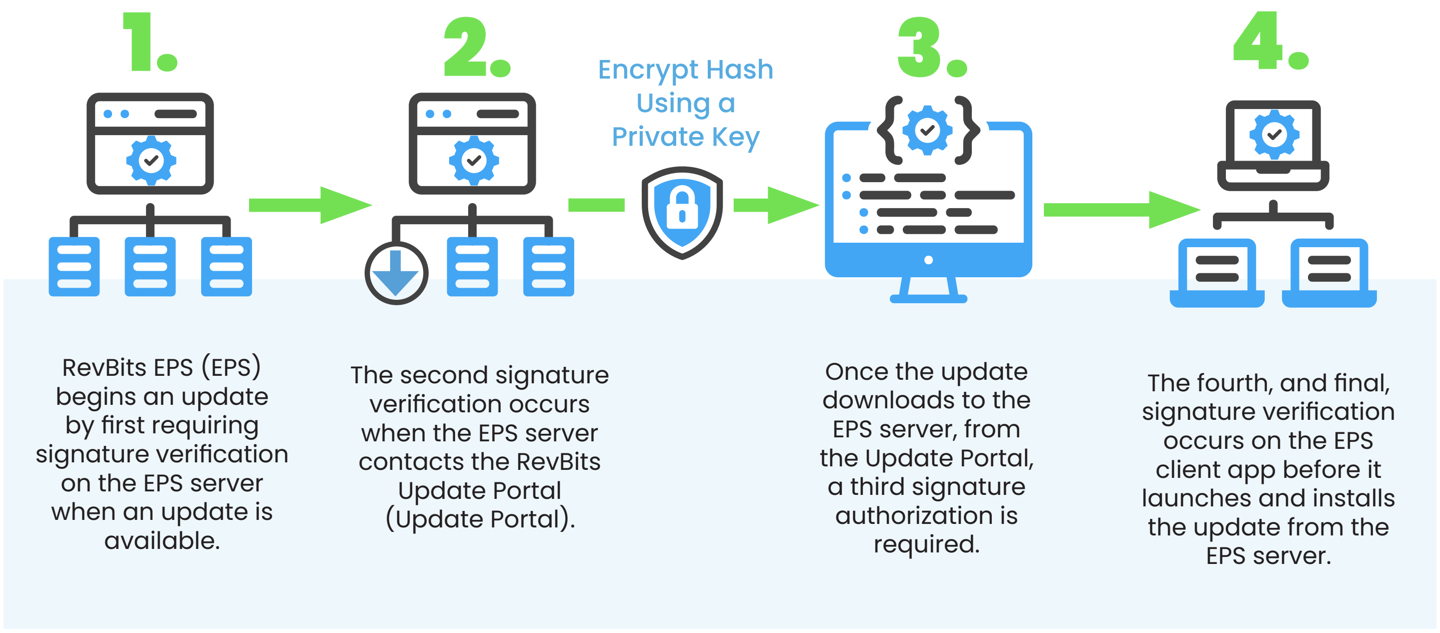
We know about the continuous flow of breaches on companies that didn't update their software in time, leaving vulnerable security holes open. More recently, we've seen supply chain attacks that target software vendors and developers, by accessing source code, build processes, and update mechanisms to distribute malware.

Unfortunately, many vendors are more concerned about simplifying build implementations and automating processes, than building strong security and implementing checks and validations at each and every step of the process. To be sure, signing each package multiple times, and verifying each step, is a cumbersome process. But it's well worth it.

Few vendors employ the discipline, integrity and rigorous procedures required to protect their software. For example, Solar Winds incorporates their product keys into the build process. This is what accounted for the SolarWinds supply chain attack, where the company's code signing procedures seemingly failed. Hackers inserted Sunspot malware into the SolarWinds system, allowing them to monitor and hijack the SolarWinds Orion platform build process. During compilation, source code file content was swapped out with a version that contained the Sunburst malware. The resulting executable was digitally signed by SolarWinds, along with Orion updates. The Sunburst malware was able to infiltrate more than 18,000 private and government networks. The breach could possibly have been mitigated if their code signing procedures had been more disciplined and diligently enforced.

RevBits rigorously secures its product builds and updates

To ensure secure protections for our software builds and updates, RevBits uses separate signing of modules, physical keys, and manual approval processes. This is important, because, while automation is great for many tedious and mundane processes, manually controlling code signing procedures is necessary, as evidenced by the SolarWinds breach.


1.
2.
Encrypt Hash Using a Private Key
3.
4.
RevBits EPS (EPS) begins an update by first requiring signature verification on the EPS server when an update is available.
The second signature verification occurs when the EPS server contacts the RevBits Update Portal (Update Portal).
Once the update downloads to the EPS server, from the Update Portal, a third signature authorization is required.
The fourth, and final, signature verification occurs on the EPS client app before it launches and installs the update from the EPS server.

RevBits EPS updates are conducted under an elevated security process, with signature verification required for every step.

A brief synopsis of how RevBits updates our Endpoint Security (EPS), which is the same process for all RevBits products.

RevBits EPS is separated into different server, client, and sub-component modules. When a new RevBits EPS version is released, we digitally sign every file, DLL, EXE, SYS, etc., with a highly protected code signing certificate. All files and final setup installer are signed with an EPS-specific private key. The associated EPS server contains the public key for validation. The EPS key package is another key required to upload and update the software, that must be separately signed. Our process receives the package, processes it, and checks the signature and hash. If everything is good, customers are notified about the new version's availability.

The server application receives and caches the update. The download update is checked using the EPS product-specific key. If the key matches, the update is accepted, and becomes available to EPS client endpoints to download the new version when they connect to the customer's EPS server. The EPS endpoints check to see if the update is properly signed, and only then will they start the installer and update. This EPS update process is just one example, as all RevBits products go through the same rigorous updating process to ensure consistent rock-solid security.

The RevBits update process is rock-solid, and can't be broken into. RevBits has a zero trust update delivery mechanism that doesn't trust any component; not the digital signature, not the update portal for delivering updates, not client's server, not the downloader.

If someone breaks into the Update Portal, can they release an update? Absolutely not!

If someone breaks into a customer's EPS server, can they release a malicious update to their EPS endpoints? Absolutely not!

If someone steals RevBits' digital security key, can they release an update? Absolutely not!

Simply put, it's impossible for hackers to release a malicious update, or pull off a supply-chain attack on RevBits products. All RevBits products have secure codes and signatures for each and every step in the process. There's no taking a chance on a flip of the coin.

Learn more about RevBits