PUBLIC SUBMISSION

As of: 4/25/22 12:44 PM
Received: April 22, 2022
Status: Pending_Post
Tracking No. l2a-vk05-0psc
Comments Due: April 25, 2022
Submission Type: Web

Docket: NIST-2022-0001
Evaluating and Improving NIST Cybersecurity Resources: The Cybersecurity Framework and Cybersecurity Supply Chain Risk Management

Comment On: NIST-2022-0001-0001
RFI-2022-03642

Document: NIST-2022-0001-DRAFT-0031
Comment on FR Doc # N/A

Submitter Information

Email: [redacted]
Organization: Virsec.com

General Comment

Thank you for the opportunity to contribute to the conversation around improving NIST guidance and enhancing its already significant impact on cybersecurity governance and NIST adoption.

* See the attached PDF for a more in-depth analysis of these proposed changes.

First, we need to define the end goal that we must achieve with this new initiative. That goal is to reach a milestone wherein the enterprise can focus on their mainstream business and not have to be overly concerned about cyber-attackers disrupting their business even if the software applications in the enterprise infrastructure have vulnerabilities.

To further define this goal, we propose the following outcomes:
- A vulnerable application is no longer an automatically exploitable application
- Security Controls should have deep visibility in the application's runtime so that they operate on facts instead of being based on presumptions of what may be happening in the application's runtime
- Protection decisions of the security control are applied automatically, with minimal to no human intervention, and in milliseconds, not in minutes, hours, or days. Delayed disposition only serves to empower the cyber attacker
- To reach the abovementioned worthy goal, NIST CSF needs to be extended to add the following three additional control families. Extending these three areas will greatly improve the cyber risk posture of an organization and thereby increase the adoption of NIST across all sectors organically driven by its effectiveness and not punitive measures.
1) Chain of Trust (CT) for software supply chain: A chain of trust tying the ISV code to the executed code must be established.
2) Runtime Visibility (RV): Protection decisions must be made on the basis of verifiable facts and not on the basis of unsubstantiated conjectures.
3) Compensating Controls (CC): Some permitted application actions may need to be restricted upstream

to achieve zero-dwell time earlier than before.

See the attached PDF for a more in-depth analysis of these proposed changes.

Thank you

Attachments

Attachment1_Runtime_Visibility

Thank you for the opportunity to contribute to the conversation around improving NIST and enhancing its already significant impact on the cybersecurity governance.

First, we need to define the end goal that we must achieve with this effort. ***That goal is an environment where cyber-attacks are irrelevant to our systems.***

To further define this goal with the following desired outcomes:

- A vulnerable application is no longer an automatically exploitable application
- Security control decisions are made based on non-repudiable facts rather than circumstantial data. This ensures protection decisions yield near-zero false positives
- Security control decisions are made in milliseconds and not minutes, hours, or days with minimal human intervention. Extended exposure helps bad actors maximize their gains.

To reach the worthy goals above, NIST CSF needs to be extended along three main points. Extending these three areas will greatly improve the cyber risk posture of an organization and thereby increase the adoption of NIST across all sectors and bring closer alignment between compliance and cyber security. Compliance and Cyber Security have historically been in two swim lanes and these changes are targeted at addressing the gaps and merging these swim lanes.

1. **Chain of Trust (CT) for Software Supply Chain:** A chain of trust tying the ISV code to the executed code must be established.

   Current state of the industry:

   Many ISVs do not publish file integrity data for the code they release in a very public way. Instead, some ISVs' sign their code using certificates to help end users establish file integrity and pristineness of the code base. Many ISVs; do not sign scripts; just executable code.

   Signing code by itself does not go far enough because certificates can be stolen or revoked unbeknownst to the end-user. Please see the recent example of where Nvidia's code signing certificate was compromised (Arntz, 2022). Other end users ensure that they check the certificate once before downloading the code. Clearly this is not enough also since certificates and be revoked after being downloaded. Yet other end users execute code blindly with no integrity checks being performed at all.

   ISVs' also do not publish the URL from where their released code can be downloaded. As a result, attackers can misdirect end users to alternate rogue locations.

   Any of these conditions can easily lead to malicious code getting downloaded, installed, and executed by the enterprise and end users much to their detriment.

   Proposed state:

   What is proposed is that in addition to signing code (including executables, libraries, scripts, etc.), the ISV must (a) release read-only file integrity information for every piece of code in each application that they release and (b) URL for where such code can be found by end users. Such information must be available publicly and securely for use by end users.

What is also proposed is an application control paradigm such that (a) the end user not only enforces file integrity information before permitting an executable to load into memory and (b) downloads the ISV code from the explicit URL stated by the ISV. This will ensure application code is properly sourced and pristine code executes every time.

- In the NIST CSF there needs to be support for creating this "Chain of Trust" between the Independent Software Vendor (ISV) that creates the software, the software distributor, and the end user using the software. NIST begins to touch on this with SSDF (SP 800-218) but the scope of these controls needs to be expanded to create a mechanism that guarantees the authenticity of the code/software for the end user at runtime.

  This is a logical extension of the NIST 800-53 SI (System and Information Integrity) control family.

  - SI-3 Malicious Code Protection needs to be extended to enforce Chain of Trust Integrity Verification Mechanisms detailed above.
  - SI-6 needs to be extended to enforce a notification back to the ISV of failed verification tests to ensure timely response to corruption in the chain trust.

2. **Runtime Visibility (RV):** Protection decisions must be made on the basis of verifiable facts and not on the basis of unsubstantiated conjectures.

   Current state of the industry:

   Imagine someone on the street outside your home trying to determine if a crime is in progress indoors? By just observing from the street, they have very little visibility or context and are very likely to make incorrect decisions about what is going on.

   Host-based security controls such as EDRs, AV, HIPS etc. operate outside the application's runtime and thanks to limited to no visibility in the application's runtime, make poor decisions that often lead to false positive alerts and often true negative which clearly do not even generate alerts. Therefore, such security controls also need human curation thereby increasing the true and total costs of securing the workload.

   Lack of visibility hurts the end users immensely and only serves to dramatically empower attackers. Attackers salivate when end users operate with security controls that lack visibility. Security controls that, instead of using data collected from within the runtime visibility, rely on analyzing network traffic for indicators of compromise (or similar pre-execution data) or rely on logs and behavioral models (or similar post-execution data) just **do not** have the precision to provide automated responses or timely protection.

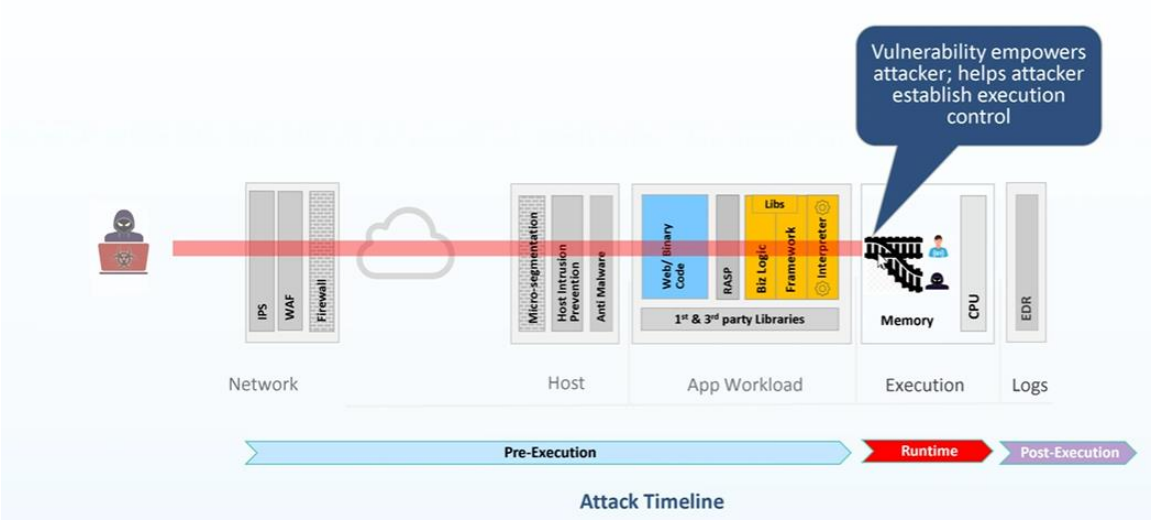
   Proposed state:

   This lack of runtime visibility, especially in applications that accept input from remote end-users, can be eliminated if the host-based security control were to have deep runtime visibility into the application's runtime. This runtime visibility is akin to having a GPS in the vehicle guiding the driver from going off the rails. The lack of runtime visibility is like looking at the map once before the journey started.

A security control that has granular runtime visibility, will make precise decisions that are not based on conjectures or extraneous considerations; instead, these decisions will be based on actual operating conditions and will be available in milliseconds.

To carry the street and home scenario described in the current state section above, this time, the human adjudicator is inside the home and bears eyewitness precision to the actual crime thereby eliminating false positives and true negatives. When this human adjudicator does not make mistakes, the cost and time to adjudicate the crime is greatly reduced. Even more importantly, the rights of the homeowner are protected and in a timely manner.

As illustrated below we can see that the only truly effective (near zero false positive detection) takes place at runtime when we can see the original intent of the code deviate from is original path as malicious data is digested by the code resulting in abnormal process branching in memory.



Runtime based controls are the only way to achieve the desired future state of cyber security. The state where there is virtually zero hacker dwell time, threats are identified and stopped in milliseconds, and there a nearly zero false positives.

Malicious code cannot run when applications are verified internally. The key points are no signature-based detection delays, no outside dependencies, and associated delays in updating code validation, and pristine code is known and controlled within the application sphere. **Runtime Visibility** is needed. To determine the difference between false positives and true positives, runtime visibility must be utilized. This will result in virtually zero hacker dwell time on systems. Meantime to Recovery (MTTR) should be smaller than the hackers' ability to achieve their malicious objective. In ideal circumstances, zero dwell time is preferable. Protection must supersede remediation.

Runtime Visibility (RV) is a further logical extension of the NIST 800-53 SI (System and Information Integrity) control family.

SI-16 Memory Protection: This control needs to be extended to include Runtime Visibility and the detection of malicious code insertion/execution within memory. The ability to stop all malicious code before it executes is the foundation of zero hacker dwell time. Current protection mechanisms detect and react after malicious code has executed and done damage.

3. **Compensating Controls (CC):** Some permitted application actions may need to be restricted upstream to achieve zero-dwell time earlier than before.

Current state of the industry:

When a new vulnerability gets reported into the vulnerability database ISVs and end users struggle to develop and deploy patches.

Vulnerabilities in application code are inevitable. During 2021, over 28K vulnerabilities were disclosed into the vulnerability databases. Of these, ~8500 vulnerabilities allowed attackers to run arbitrary code (think malware) on the victim's workload. Even more importantly, ~5800 vulnerabilities had exploit code posted on the Internet within a day of the vulnerability being disclosed. The ease with which a vulnerability can be exploited easily empowers even rookie attackers.

It is a well-known fact that an unprotected workload that is accessible over the Internet will be attacked within 7 minutes.

It is also a known fact that most organizations just cannot patch in 7 minutes or less. So what does the end-user do to protect themselves?

Proposed state of control

It would be ideal if an application which may be replete with vulnerabilities should not be allowed to execute certain operations as further upstream as possible. We can achieve this mission by putting compensating controls in place. Such compensating controls should not only prevent vulnerable code from being exploited but should also facilitate remediation of flaws by the ISVs developers.

This effectively means an otherwise trusted code needs to be restricted from performing certain specified operations. To implement such a control, enterprise IT must be able to specify an "exception" to the allow-listing application policy proposed in the Chain of Trust control. Conceptually this is a lot like the firewall which by default locks down an application but in which a hole must be punched for legitimate network activity of an application to occur. To accomplish such a scenario, we propose an Application Control "firewall" that restricts certain invocations of an otherwise permitted applications to be specified by the end-user.

References

Arntz, P. (2022, March 15). *Stolen Nvidia certificates used to sign malware-here's what to do*. Malwarebytes Labs. Retrieved April 15, 2022, from https://blog.malwarebytes.com/awareness/2022/03/stolen-nvidia-certificates-used-to-sign-malware-heres-what-to-do/#:~:text=Leaked%20Nvidia%20certificates,them%20to%20sign%20their%20malware.

Bell, S. (2015, April 27). *How hackers access your computer*. BullGuard. Retrieved April 15, 2022, from https://www.bullguard.com/blog/2015/04/how-hackers-access-your-computer.html