Waratek Name Space Layout Randomization FAQ

(January 25, 2017)

Q. What is Name Space Layout Randomization (NSLR)?

A. NSLR is a security feature invented by Waratek that is available with the Waratek Application Security Platform. NSLR is based on the same principle as Address Space Layout Randomization ASLR. Both NSLR and ASLR provide protection against code injection attacks.

NSLR hardens the JVM by randomizing the JDK namespace (Java packages), which makes code injection exploits so difficult to execute that they become unfeasible. NSLR protects against known and unknown, zero day exploits. The current release of Waratek NSLR only protects the host Java Virtual Machine (JVM), but future releases will also protect the guest JVC (Java Virtual Container).

Q. How does NSLR work?

A. In Java, an exploit author that implements a Code Injection exploit needs to know the exact names of classes and packages that must be invoked. This is instrumental in executing a successful attack. Making the Java class packages non-deterministic makes any exploit unsuccessful. In effect, it deactivates any code injection exploit.

For example, with NSLR enabled, the *java.lang.System* class will be randomized and renamed to something like *$85rbuLjHNERijUhN.lang.System*. Any exploit that tries to invoke *java.lang.System* will fail. For attackers to successfully execute an attack they would need to know the randomized package name. In addition, the randomized package name is different every time the JVM boots.

Also, it is impossible to access the randomized host package name from the guest JVC, since these are not visible to it.

Attempts to brute force the system and retrieve the randomized package name is not feasible in a reasonable amount of time since NSLR uses 96-bit names, which would likely require several thousand years to crack the encryption. The randomized bits are configurable up to 1024 bits.

Finally, a code injection attack performed in an NSLR enabled JVM will generate a Class Not Found Exception. Currently, these events are not identified and logged. A future release will identify when a code injection attack was attempted and create a security event log.

Q. How is NSLR different from Address Space Layout Randomization?

A. Both NSLR and ASLR are protection mechanisms against code injection attacks and do not remove vulnerabilities. ASLR is a feature of the Operating System, while NSLR is a feature of the Waratek Application Security Platform. ASLR makes addresses difficult to resolve because it randomizes them. NSLR makes java packages (namespaces) difficult to resolve because it randomizes them.

Q. How do organizations currently protect against code injection attacks?

A. Code injection attacks exploit specific vulnerabilities. The most common approach to protecting against them is to identify and fix/patch vulnerabilities. If a vulnerability cannot be mitigated, the application functionality/service associated with the vulnerability can be disabled or removed. This is not ideal, and may or may not be feasible. Alternatively, a Firewall, IPS or WAF can be used to try to mitigate code injection attacks, however they are often plagued by false positives.

Q. How common are code injection attacks?

A. Code injection attacks rank 6th among 2055 known vulnerabilities according to MITRE.

Q. What are some examples of well-known attacks/breaches attributed to code injection attacks?

A. Click for a list of known code injection attacks.

# # #