RunSafe Security and Memory Threats: Using Alkemist® to Fight the Exponential Growth in Memory Vulnerabilities and Immunize Software from Attacks

TABLE OF CONTENTS

EXECUTIVE SUMMARY

Over the last two years, memory-based attacks have become the largest, most damaging, and fastest-growing class of cyber threats. MITRE flagged memory vulnerabilities as by far the highest-ranking software weakness in their 2019 Common Weakness Enumeration (CWE)[1]. In an article reporting on the 2019 RSA Conference, privacy and security journalist Byron Achihido observed, “if you examine any high-profile data breach, you’re likely to find memory-hacking techniques utilized at multiple key stages of the attack.”[2]

Software vendors, cybersecurity firms, and IT departments are all now working – and investing – aggressively to detect and patch/remediate memory vulnerabilities. Huge chunks of IT budgets have been redirected to combat these threats, but existing defense in depth methods were not designed to solve this problem. Organizations are putting more and more extensive scanning and patching in place to hold back memory hacks, draining budgets and delaying time to market for critical digital initiatives. But the problem still persists, and damage is still happening.

With each new release, developers introduce new weaknesses and errors in software code that create new memory vulnerabilities. Even the best software companies, such as Microsoft, Amazon, and Google, still produce software code with errors, and most companies are not as good. Beyong missing existing vulnerabilities, the S/DAST and scanning tools being put in place to mitigate this problem slow down developers, creating an expensive drag on software development methodologies. In today’s world of continuous development and deployment, companies must find effective mitigation technologies that “shift left” to integrate into software development without slowing down developers.

Expert cybersecurity researchers and developers have created new Moving Target Defense (MTD) and Runtime Application Security Protection (RASP) approaches that shift hacker economics back in favor of the defenders. Several efficient, cost-effective, and proven technologies can now proactively protect your strategic software and firmware assets.

This white paper covers:

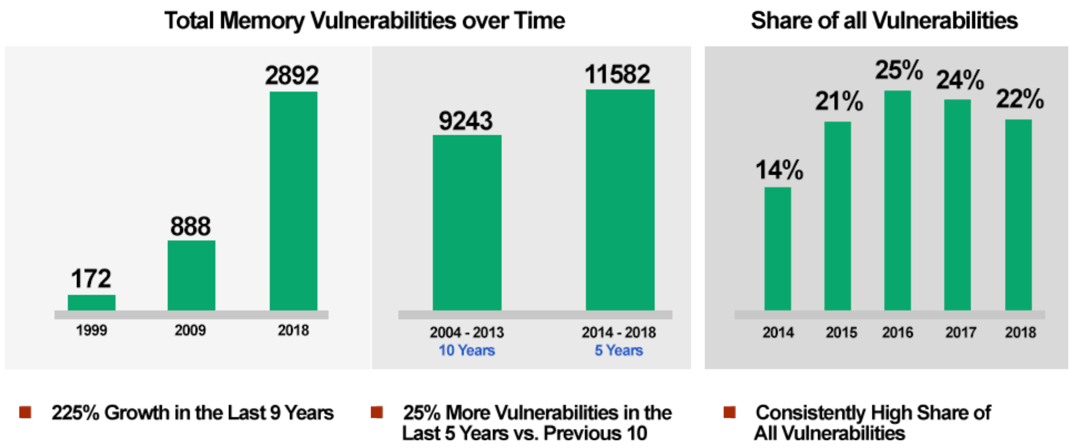
1. Market data on the scope of the memory threat problem
2. Limitations of existing cybersecurity Defense in Depth approaches and tools
3. Effective MTD and RASP approaches to build security protection into code

RunSafe created the Alkemist® transformation engine to bring a new cyberhardening approach to market at scale and quickly give enterprises a solution to a wide range of memory vulnerability exposures.

Scope of Memory Vulnerabilites

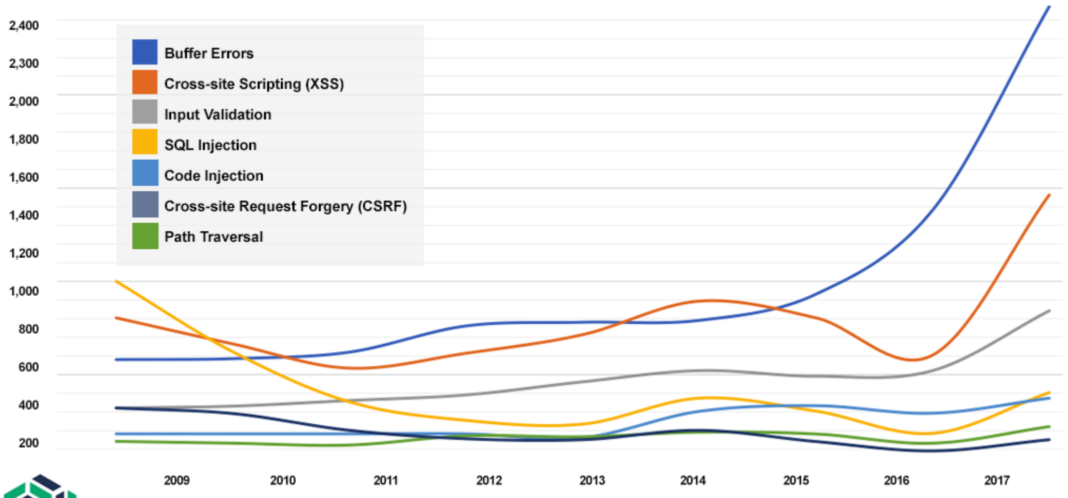
Memory vulnerabilities have tripled over the last nine years, with the discovery rate of new memory vulnerabilities more than doubling over the last five years versus the prior ten, according to NIST CVE data.[3] Buffer errors (NIST’s term for memory vulnerabilities) became the largest vulnerability class in the National Vulnerability Database in 2015, grew exponentially, and in 2017 recorded over 1000 more instances (2400+ vs 1400+) than the second-biggest weakness.

Memory Based Threat Landscape



NIST CVE Data Source: https://www.cvedetails.com/vulnerabilities-by-types.php

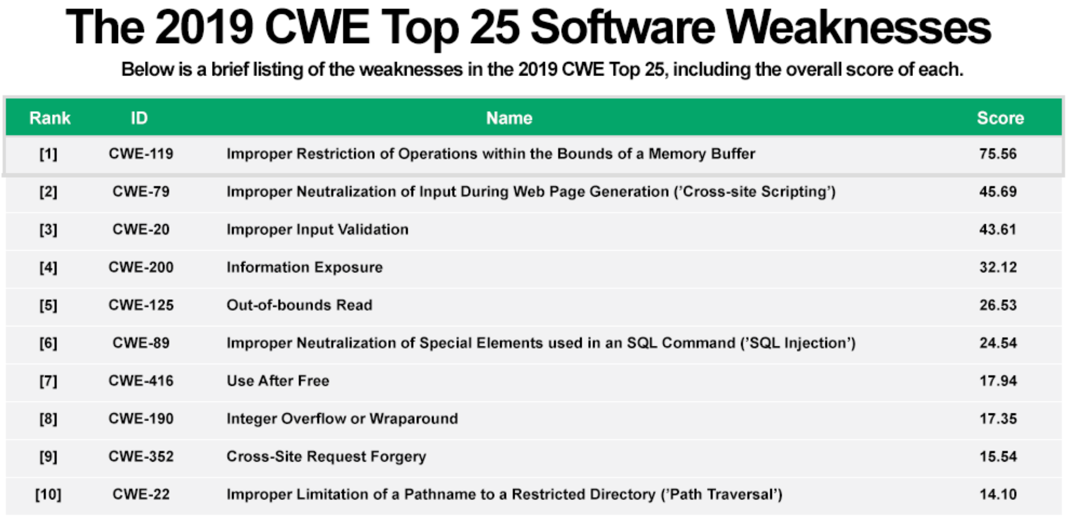
Memory Vulnerabilities Growing Exponentially



Source: NIST National Vulnerability Database (NVD)

• 1775 Tysons Boulevard, Fifth Floor, McLean, VA 22102 USA
runsafesecurity.com • sales@runsafesecurity.com • +1 (571) 441-5076

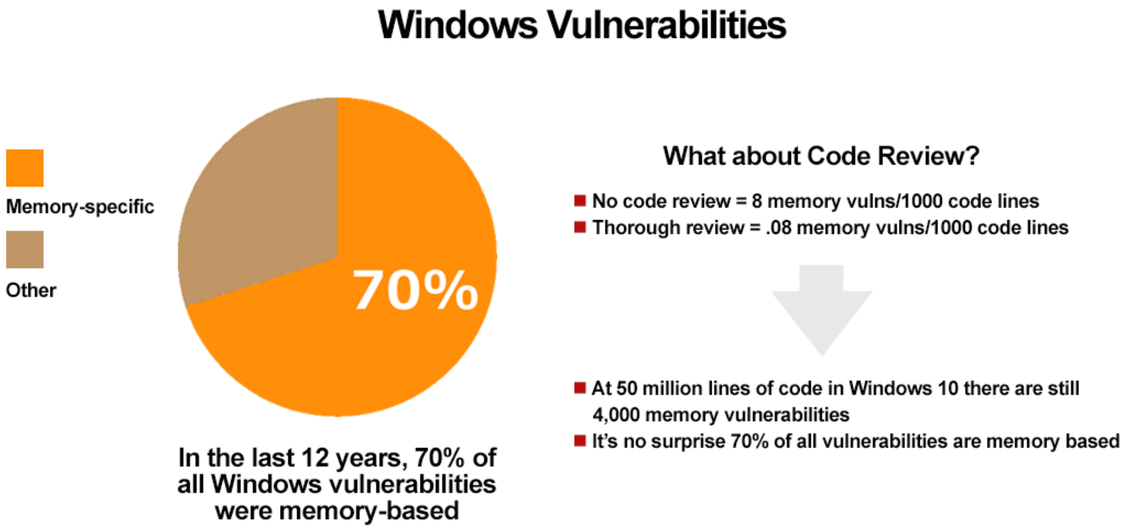
In September 2019, MITRE ranked memory vulnerabilities the most dangerous weakness in software, with a total vulnerability score of 75.56 out of 100 -- almost 30 points higher than any other weakness.[5] The 2018 Ponemon Institute State of Endpoint Security Risk study predicted that these “file-less” techniques will account for 38% of targeted attacks in 2019.[6] Trend Micro’s Roundup Report for the first half of 2019 validates that projection, showing file-less attacks up 265% over the same period a year ago.[7] And earlier this year, Microsoft revealed that over 70% of all security bugs over the last 12 years have been memory weaknesses.[8]



The 2019 CWE Top 25 Software Weaknesses

Below is a brief listing of the weaknesses in the 2019 CWE Top 25, including the overall score of each.

Rank	ID	Name	Score
[1]	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer	75.56
[2]	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	45.69
[3]	CWE-20	Improper Input Validation	43.61
[4]	CWE-200	Information Exposure	32.12
[5]	CWE-125	Out-of-bounds Read	26.53
[6]	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	24.54
[7]	CWE-416	Use After Free	17.94
[8]	CWE-190	Integer Overflow or Wraparound	17.35
[9]	CWE-352	Cross-Site Request Forgery	15.54
[10]	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	14.10



Windows Vulnerabilities

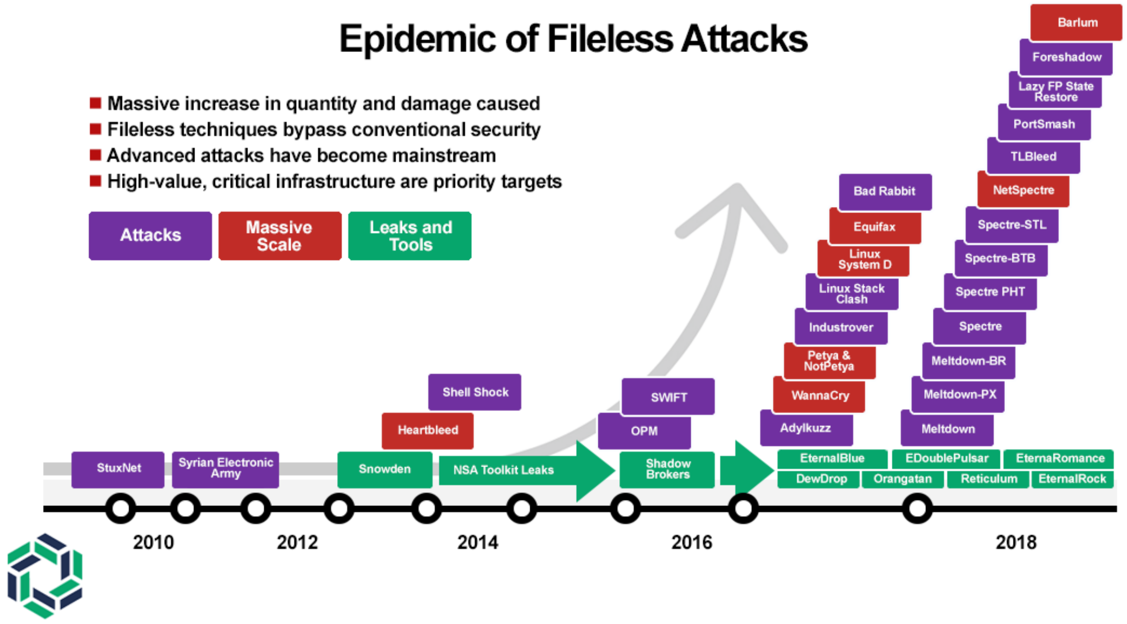
In the last 12 years, 70% of all Windows vulnerabilities were memory-based

What about Code Review?

- No code review = 8 memory vulns/1000 code lines
- Thorough review = .08 memory vulns/1000 code lines

- At 50 million lines of code in Windows 10 there are still 4,000 memory vulnerabilities
- It’s no surprise 70% of all vulnerabilities are memory based

These so-called “file-less” attacks have emerged as the fastest-growing threat vectors since 2017’s launch of the Stuxnet-based WannaCry virus. Many other common attacks, including phishing/social manipulation and zero-day worms, are also taking advantage of memory exploitation to multiply the duration and scope of the damage they inflict. The following chart shows the dramatic rise of memory exploits over the last 2 years, and the pace has continued to accelerate.



Beyond the sheer prevalence of vulnerabilities in software, it’s important to consider the broader implications that code vulnerabilities present for enterprises. A few of the most serious include:

• Supply Chain Concerns - If we consider the complex global supply chain that exists for many enterprises, there are numerous opportunities for attackers to infiltrate a finished product. In many cases, a software bill of materials (BOM) includes packaged code from suppliers all over the world. Not all of the suppliers may have the same level of cybersecurity hygiene required by the final producer. This problem is more pronounced when you consider the hundreds or thousands of vendors an enterprise manages, increasing the attack surface and making points of entry easier for an attacker.

• Addressing Scaled Attacks - Most IT, OT, and IoT systems are made up of software components found in many enterprises. Customer Relationship Management (CRM), Enterprise Resource Planning (ERP), Supply Chain Management (SCM)...choose your acronym for whichever enterprise system you prefer, and it’s likely deployed globally at thousands of companies on many thousands of systems. If an attacker is able to find a way in, that vector can then become the access point into all other environments within the same system. Or, if it’s a device deployed hundreds or thousands of times in a single enterprise environment, one vulnerability exploited on a single device can quickly expand into a system-wide attack with dire consequences.

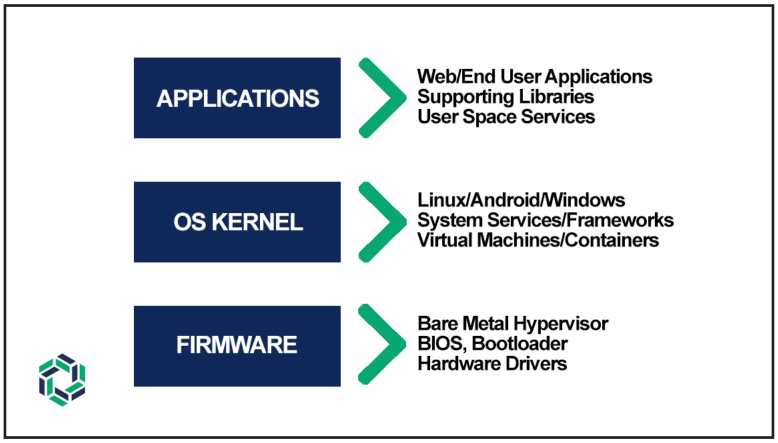
- Virtualized System Exploitation - As the move to virtualized environments continues, the underpinnings of these systems introduce new attack surfaces. Both hypervisor and Virtual Machine (VM) instances provide the “plumbing” to a virtualized environment and are a valuable beachhead to an adversary who seeks to move far and wide across an organization. VM escape exploits are increasingly an area of significant risk given the widespread damage they inflict with the ever-increasing move to the cloud. Cohabitation in the cloud also can allow an attack against one company to bleed across to another company’s assets via a VM escape.

These are just three examples of how the changing threat landscape forces a new approach to securing enterprise Systems. In addition, OT is at increased risk given its tighter integration with traditional IT networks that expose new attack vectors. At the edge, IoT represents a massive footprint with a low dollar per unit cost that constrains the ability to apply expensive hardware-based security mechanisms. Attack surfaces have changed, attack methods have evolved, and a new approach is required.

EFFECTIVENESS OF DEFENSE-IN-DEPTH IN TODAY'S THREAT LANDSCAPE

Cybersecurity best practices most frequently employ a "defense-in-depth" architecture. That is, one that ensures that the perimeter is protected, authentication is strong, data is encrypted, and integrity is preserved throughout the system. Key components include firewalls, VPNs, identity access management, intrusion detection, application security testing, on-device encryption, and anti-virus, among other emerging tools and techniques. It frequently also includes organizational training that equips workers with the minimal skills and awareness necessary to prevent common attack methods like phishing and social engineering. Unfortunately, none of these tools are effective against many of the file-less attack types detailed above.

This problem is expanding exponentially because the number of connected devices is growing rapidly, and they are being attached to a virtualized environment that no longer sits behind a corporate firewall. This phenomenon, in combination with long, distributed supply chains, is increasing attack surfaces significantly.



Software flaws that create memory vulnerabilities are found across the board at all levels of the traditional IT/OT stack:

Such a vast attack surface puts the enterprise at a distinct disadvantage. It must ensure system integrity across its entire IT, OT, and IoT footprint, while an attacker only needs to find one weakness to take it all down.

Defenders must adapt to this changing landscape and embrace new approaches to secure valuable IT, OT, and IoT assets that store critical data and drive essential business processes.

Regardless of how "in-depth" a cybersecurity program is, persistent attackers with the means and motives will always find a way through it. Until recently, the best defensive option has been to monitor for the presence of an attacker, and should one be detected, work as quickly as possible to isolate the issue, limit the damage, remediate the breach and restore normal operations: the infamous "whack-a-mole" strategy. Many organizations have now recognized the gap in software code protection and adopted aggressive code hygiene regimes, designed to scan for known vulnerabilities and apply patches as soon as they are available. These reactive approaches consume scarce resources, extend implementation timelines, and still leave exploitable gaps for sophisticated cyber criminals.

MEMORY PROTECTION FROM FILE-LESS ATTACKS

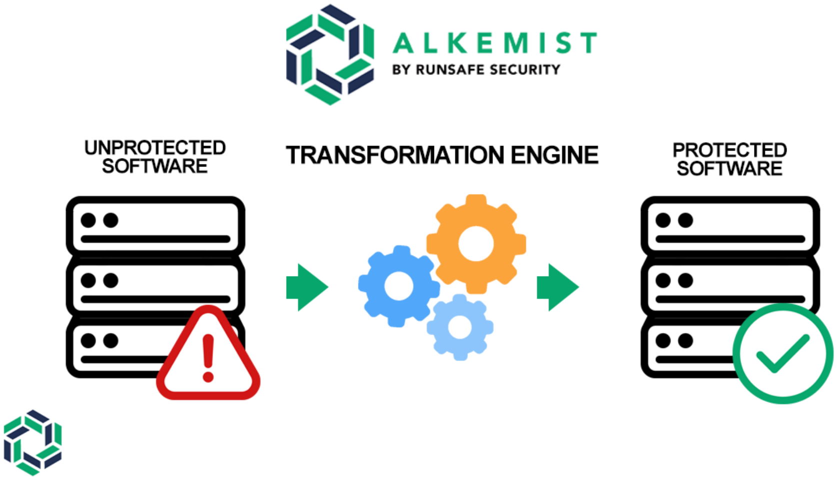
A new class of proactive cybersecurity solutions now allows organizations to solve this problem by building protections directly into the systems themselves. Runtime Application Security Protection (RASP) and Moving Target Defense (MTD) approaches enable Information Technology (IT), Operational Technology (OT), and Internet of Things (IoT) systems to protect themselves from attack. These techniques defend against memory corruption errors and buffer overflows, some of the most damaging zero-day attacks, which currently represent between 30 and 50% of all Common Vulnerabilities and Exposures[9].

When an attacker breaches traditional cyber protection layers, RASP and MTD technologies inside the target system, embedded in the code itself, prevent the attack from the inside out. This approach is viable across all layers of the code stack from bare metal firmware, to the operating system (OS) and up through the application layer, across a wide array of IT, OT, and IoT environments. A number of RASP/MTD based approaches are now available, but few are truly applicable across a wide variety of hardware and software environments.

RUNSAFE ALKEMIST SOLUTION—AUTOMATED CYBERHARDENING

RunSafe's Alkemist software-based transformation engine can apply RASP/MTD protections directly into code, enabling that code to protect itself from common attack methods. Specifically, this solution protects against file-less attacks seeking to exploit memory corruption bugs. The protections applied are called "transforms," and we refer to the process as "cyberhardening." Alkemist implements several different effective transform types, each of which has unique properties that, when applied, allow software code to protect itself from attack.

The basic workflow is depicted below:



The Alkemist transformation engine can apply three types of protection:

- LFR – Load-time Function Randomization
- BBR – Basic Block Randomization
- SFR – Stack Frame Randomization.

Each transform can be applied to a target system independently, standing on its own as a powerful protection from attack. However, they are exponentially more powerful when applied together. Such transformations are ideal for cyberhardening open source software, in-house developed code, or 3rd party binaries. In the next section we explain how each of the transforms works.

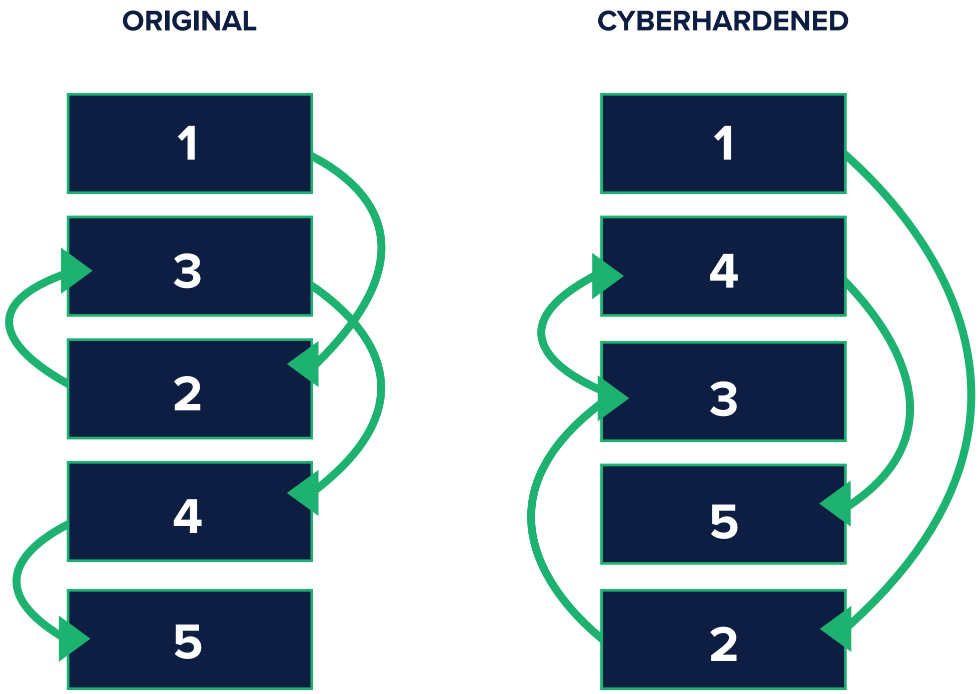
MEMORY CORRUPTION BUG PROTECTION (LFR & BBR)

Memory corruption bugs specifically allow attackers to exploit the way functions in software are loaded in memory. One of the core constructs attackers rely on to leverage these types of bugs is uniformity. In normal operations, functions execute using a consistent memory structure in which memory address locations are consistent every time a function is called. When an attacker is looking to exploit a vulnerability, they rely on this consistency to ensure that they can find the building blocks necessary to carry out a reliable attack.

The Alkemist® transformation engine removes this uniformity and introduces randomness. Address locations are no longer consistent, and even if the attacker were able to access a system and find a vulnerability, that attack would not work. This approach randomizes memory address locations so the attacker cannot reuse existing code for their exploit.

See how this works below:

LFR, BBR



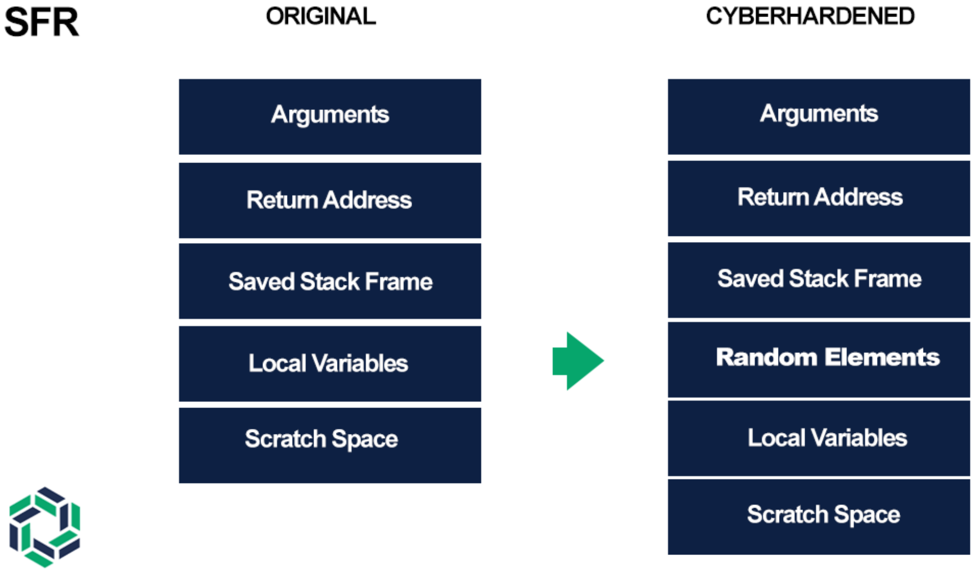
There are two highly effective technical methods to implement this type of randomization: Load-time Function Randomization (LFR) and Basic Block Randomization (BBR). LFR and BBR behave in similar ways as depicted in the figure above. The key difference is that LFR will randomize each “whole” function in memory across an entire binary when it is executed/loaded, while BBR will randomize, once per transform, the basic blocks that comprise an individual function. Both approaches remove consistency, taking away the underlying vulnerability that hackers are seeking to exploit.

PROTECTION FROM STACK BASED ATTACKS (SFR)

Stack-based overflows are a time-honored vulnerability in the adversary community. Like other memory corruption bugs, they are dangerous and prevalent. These bugs are found when local variables or data need to live temporarily on the “stack” within the processing environment and a user can control how much data is written to the stack. When data is written to the stack, it is always written to the same location relative to the start of that function’s stack. Like other memory corruption exploits, attackers rely on a consistent location and structure to know where the function’s return address is on the stack. Once they find a buffer overflow, they can easily compute how many bytes to write to carry out their attack successfully.

The “Stack Frame Randomization” (SFR) technique eliminates predictability of the return address on the stack. SFR-protected code places a randomly-sized offset onto the stack, thereby changing the location of all data on the stack during execution.

See how this works below:



With SFR protections in place, an adversary attempting to exploit a buffer overflow vulnerability will not be able to locate the return address on the stack correctly to launch their attack. SFR removes the underlying construct, consistency, thereby preserving system integrity.

ROP/JOP PROTECTION

Another target area for attackers is control flow. When software runs on a system, it's typically running a number of different functions in sequence. In some cases, the code is written to ensure that all the functions are securely identified and cannot be run out of sequence. Other times, there may be functions that aren't securely addressed and could potentially be exploited to redirect the expected sequence. This is commonly referred to as a ROP/JOP attack (Return or Jump Oriented Programming).

In this type of an attack, the adversary is able to redirect the flow of function execution to an arbitrary function of the attacker's choosing. If this occurs, the attacker can effectively take control of the system and execute their own payload or malware. LFR and BBR protect against ROP/JOP attacks by moving the location of ROP/JOP gadgets in memory (by moving the code containing those gadgets).

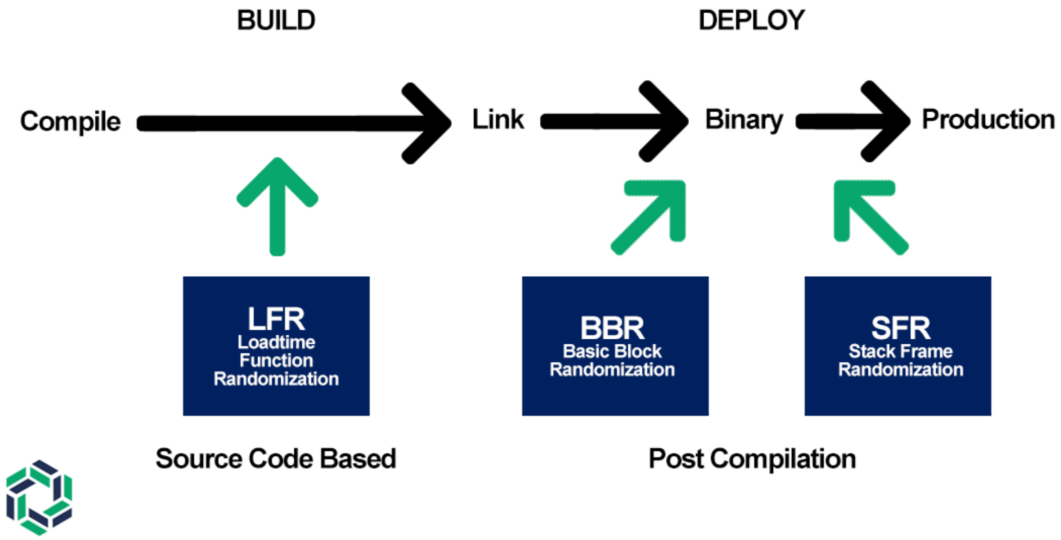
IMPLEMENTING ALKEMIST

Using an effective transformation engine to automatically implement these protections completely eliminates the attacker's underlying construct for this kind of exploit. They are no longer able to redirect code execution to the payload of their choosing. An implementer has the freedom to apply each of these powerful transforms to software in different forms from different sources. Examples such as a virtualization instance running an open source KVM hypervisor, in-house developed code for a custom application, or 3rd party binaries as part of a complex electronic device are all in scope.

As depicted below, Alkemist can apply the transforms at different times during the software compilation process:

• Source Code Based - LFR is a "source-based" transform that is applied to code before a final compiled binary is produced. It's important to note that LFR does not modify any source code. In fact, protections are applied to assembly code, during compilation, thus enabling automated hardening of all compiled code without a developer having to manually apply the protection.

• Post Compilation Based - BBR and SFR are considered "post-compilation based" transforms in that they are applied to code after final compilation. This offers unique benefits for enterprises that incorporate binary code from 3rd parties in their systems. As access to source code isn't always possible, SFR & BBR offer strong protections even when only the binary is available.


BUILD
DEPLOY
Compile
Link
Binary
Production
LFR
Loadtime Function Randomization
BBR
Basic Block Randomization
SFR
Stack Frame Randomization
Source Code Based
Post Compilation

Conclusion

The number of connected devices is growing rapidly, and they are being attached to a virtualized environment that no longer sits behind a corporate firewall. This phenomenon, in combination with long, distributed supply chains, has caused attack surfaces to explode in size.

Defenders must adapt to this changing landscape and embrace new approaches to secure valuable IT, OT, and IoT assets that store critical data and drive critical business processes. Recent innovations and the availability of production-ready transformation engines now provide an efficient and scalable option to shift cyber economics in favor of defenders.

RunSafe's Alkemist transformation engine makes it possible for enterprises to implement code hardening quickly across the widest range of software and firmware assets, with the least disruption to their software development and deployment processes.

SOURCES

1. MITRE 2019 CWE cwe.mitre.org/top25
2. https://www.lastwatchdog.com/my-take-memory-hacking-arises-as-favored-new-tactic-to-carry-out-deep-persistent-incursions/
3. NIST CVE database https://www.cvedetails.com/vulnerabilities-by-types.php
4. NIST National Vulnerability Database
5. Source 1
6. 2018 Ponemon Institute State of Endpoint Security Risk Study https://www.ponemon.org/news-2/82
7. Trend Micro Roundup Report 1st Half 2019 www.trendmicro.com/.../threat-reports/roundup
8. https://www.zdnet.com/article/microsoft-70-percent-of-all-security-bugs-are-memory-safety-issues/
9. CVE Data Source: https://www.cvedetails.com/vulnerabilities-by-types.php

ABOUT RUNSAFE SECURITY, INC.

RunSafe Security's mission is to painlessly immunize all software from cyber attacks by disrupting hacker economics. Our security techniques inoculate our customers' systems from an entire class of cyber attacks. We integrate across build and deploy tools chains without developer friction protecting open source, in house code, and 3rd party binaries. Headquartered in McLean, Virginia, with an office in Huntsville, Alabama, RunSafe Security's customers span the DevSecOps, critical infrastructure, IIoT, automotive, medical, and national security industries.

RunSafe SECURITY

www.runsafesecurity.com
571.441.5076
sales@runsafesecurity.com