Cyber Moving Targets

Yashar Dehkan Asl

Introduction

An overview of different cyber moving target techniques, their threat models, and their technical details.

Cyber moving target technique:

- Defend a system
- Increase the complexity of cyber attacks
    * Less homogeneous
    * Less static
    * Less deterministic

Moving Target Techniques

1. **Dynamic Runtime Environment**
   - **Address Space Randomization**
   - **Instruction Set Randomization**

1. **Dynamic Software**

2. **Dynamic Data**

3. **Dynamic Platforms**

4. **Dynamic Networks**

Address Space Randomization

Address Space Layout Permutation

Threat Model:

Attack Techniques Mitigated: Code Injection and Control Injection

Defends against buffer overflow attacks

Description:

Performs stack randomization at both the user and kernel levels

Machine running programs are protected from code or control injection

Cont.

DieHard

Threat Model:

Attack Techniques Mitigated: Code and Control Injection

protects the heap from indirect buffer overflow attacks

Description:

DieHard attempts to defend against four classes of vulnerabilities that could lead to program crash or code/control injection: invalid frees, buffer overflows, dangling pointers, and uninitialized reads.

Cont.

Instruction Level Memory Randomization

Threat Model:

Attack Techniques Mitigated: Code and Control Injection

Defends against buffer overflow attacks on the stack and heap from an adversary that can provide arbitrary input to a vulnerable program.

Description:

Randomizes both the stack and heap. The randomization takes the form of a program that transforms an executable into a randomized version that has the same behavior.

Cont.

Operating System Randomization

Threat Model:

Attack Techniques Mitigated: Code and Control Injection

Attempts to defend against buffer overflow attacks through stack randomization as well as decrease the likelihood of injected code successfully running through library and system call randomization.

Description:

The authors use three different techniques to add randomness to the program environment: stack randomization, system call randomization, and movement of libc

Cont.

Function Pointer Encryption

Threat Model:

Attack Techniques Mitigated: Code and Control Injection

Defends against control injection through indirect buffer overflow attacks on the heap

Description:

Prevent indirect buffer overflow attacks by making it difficult for the attacker to overwrite a function pointer with a chosen value.

Instruction Set Randomization

G-Free

Threat Model:

Attack Techniques Mitigated: Control Injection

Mitigate ROP attacks against executables compiled with the modified compiler.

The first step to stopping ROP is eliminating all misaligned free branch instructions.

The second protection mechanism used is a careful encryption of the return pointer on the stack.

Cont.

Practical Software Dynamic Translation

Threat Model:

Attack Techniques Mitigated: Code Injection

Protects against code injection into running binaries from all vectors

Description:

This scheme “slow execution” problem by using a very lightweight virtual machine, and the weak encryption function problem by switching to AES for encryption.

Cont.

RandSys

Threat Model:

Attack Techniques Mitigated: Code and Control Injection

Defends against code injection and control injection from buffer overflow attacks on the stack and heap.

Description:

For ISR, it implements system call randomization between user space and kernel space.

For ASLR, it implements library re-mapping and function randomization.

Cont.

Randomized Instruction Set Emulation

Threat Model:

Attack Techniques Mitigated: Code Injection

This method is targeted at stopping external binary code injection into an executing program.

Description:

It scrambles the instruction set at load-time and descrambles them at runtime.

Cont.

SQLRand

Threat Model:

Attack Techniques Mitigated: Code Injection

Aims to protect against SQL injection attacks in situations where the query depends partially on untrusted input.

Description:

The SQL language is randomized so that any code that was injected will not run.

Cont.

Against Code Injection with System Call Randomization

Threat Model:

Attack Techniques Mitigated: Code Injection

Protects against injection of code into an application with a buffer overflow vulnerability.

Description:

Every system call number is replaced by a randomly chosen pseudonym.

Dynamic Software

Software Diversity Using Distributed Coloring Algorithms

Threat Model:

Attack Techniques Mitigated: Code Injection

Reduces the number of machines an attacker can successfully compromise in a network using code injection attacks.

Description:

This meta-technique involves taking existing code diversity techniques and applying them across an entire network.

Cont.

Security Agility for Dynamic Execution Environments

Threat Model:

Attack Techniques Mitigated: Exploitation of Trust

Aims to mitigate system and network intrusions at a high level by dynamically modifying security policies.

Description:

The authors describe and implement a software toolkit that allows applications to be developed around the idea of dynamically changing security policies.

Cont.

Proactive Obfuscation

Threat Model:

Attack Techniques Mitigated: Code Injection and Control Injection

Aims to mitigate buffer overflows and other injection attacks on network visible services.

Description:

Creates multiple copies of each service executable, randomized differently. The randomization used can be any of the other executable randomization techniques we have described such as ISR, ALSR, or system call randomization

Cont.

Program Differentiation

Threat Model:

Attack Techniques Mitigated: Code Injection and Control Injection

This technique mitigates buffer overflow attacks on remote services.

Description:

The authors aim to design a secure mobile phone platform that is not vulnerable to remote attack through buffer overflow exploits.

Cont.

Reverse Stack Execution in a Multi-Variant Execution Environment

Threat Model:

Attack Techniques Mitigated: Code Injection

Detects buffer overflows on the stack and prevents exploitation of them through stack smashing.

Description:

The authors propose a very simple form of multi-variant execution with two replicas where one replica runs with the stack growing upwards and the other runs with the stack growing down.

Dynamic Data

Data Diversity Through Fault Tolerance Techniques

Threat Model:

Attack Techniques Mitigated: Resource

This technique was not designed to fight malicious input directly but it is more focused on unintentional faults.

Description:

Aims to increase the fault tolerance of an application by reevaluating the input to a program using a different algorithm.

Cont.

Redundant Data Diversity

Threat Model:

Attack Techniques Mitigated: Resource and Code Injection

Aims to help mitigate attacks that target specific data inside of an application by way of malicious input.

Description:

This technique is a variation of the N-variant programming technique. In involves running multiple copies of a program that each run transformations of the original data being protected without having to rely on secrets.

Cont.

Data Randomization

Threat Model:

Attack Techniques Mitigated: Code Injection and Control Injection

Helps protect against code injection attacks by randomizing any code injected into the program.

Description:

This is a compiler-based technique that provides probabilistic protection by randomizing all the data that it stores in memory.

Cont.

End-to-End Software Diversification

Threat Model:

Attack Techniques Mitigated: Code Injection and Exploitation of Authentication

This technique has the potential to defend against different levels of code injection as well as some authentication attacks.

Description:

The idea of this technique is to compose many different randomization methods and apply them to aspects of a service that does not affect the functionality of the program.

Dynamic Platforms

Security Agility Toolkit

Threat Model:

Attack Techniques Mitigated: Exploitation of Trust

Helps mitigate the damage that can be done on a system by restricting the access an application or process currently holds in the event of attack detection.

Description:

Provides a toolkit to wrap around executables. It allows the injection of greater access control mechanisms with the ability to change them during program runtime.

Cont.

Genesis

Threat Model:

Attack Techniques Mitigated: Code Injection and Control Injection

Defends against different threats depending on how it is implemented. If it is implemented with ISR, it can defend against code injection attacks.

Description:

This technique involves applying runtime software transformations to a program. The program is run in an application-level VM called Strata.

Cont.

Multi-Variation Execution

Threat Model:

Attack Techniques Mitigated: Code Injection

Combats code injection attacks by having each running variant use a different system call mapping and unpredictable stack direction.

Description:

Involves running multiple variations of the same program. A separate monitoring program monitors all variations. The level of monitoring can vary from each program having the same result down to checking each instruction executed.

Cont.

Diversity Through Machine Descriptions

Threat Model:

Attack Techniques Mitigated: Code Injection

This technique is meant to mitigate mass code injection attacks. Each system would potentially need their own custom exploit to work because of all the varying system modifications and configurations.

Description:

Involves using a VM and compiler machine descriptions to create a diverse set of architectures.

Cont.

N-Variant Systems

Threat Model:

Attack Techniques Mitigated: Code Injection and Control Injection

The instruction set tagging variant gives each running variant their own instruction set. Since each variant is passed the same input, this will help mitigate code injection attacks because the attack might succeed on one variant but would presumably fail on another.

Description:

The idea behind this technique is to run multiple variants of the same application simultaneously without relying on anything to be secret.

Cont.

Trusted Dynamic Logical Heterogeneity System

Threat Model:

Attack Techniques Mitigated: Code Injection, Control Injection, Scanning, and Supply Chain

This technique can help mitigate a OS and architecture dependent attacks. Since the application is migrating between systems with different libraries, architectures, and layouts, it is more difficult to construct exploits that will work under every platform.

Description:

The Trusted dynAmic Logical hEterogeNeity sysTem (TALENT) is a technique that involves making a running application migrate between different platforms while preserving the state of that application.

Cont.

Intrusion Tolerance for Mission-Critical Services

Threat Model:

Attack Techniques Mitigated: Resource

This technique combats resource attacks such as DoS and data integrity attacks. It mitigates the impact of DoS attacks by trying to ensure there are enough resources on a platform to run the service.

Description:

Aims to make critical web services more survivable in the face of attack.

Cont.

Generic Intrusion-Tolerant Architecture for Web Servers

Threat Model:

Attack Techniques Mitigated: Code Injection, Control Injection, and Scanning

Helps reduce the attack surface of the services by not making them directly accessible from the outside, limiting the types of traffic that can reach it, and running on multiple diverse systems.

Description:

Aims to be a system capable of diagnosing issues, repairing itself, and reconfiguring itself in order to continue to provide a service in the event of attack.

Cont.

Self-Cleansing Intrusion Tolerance

Threat Model:

Attack Techniques Mitigated: Code Injection and Control Injection

This technique does not detect any attacks but assumes the system is continually under attack.

Description:

The self-cleansing intrusion tolerance (SCIT) technique aims to decrease the exposure time of a system by rotating it with copies.

Cont.

Genetic Algorithm for Computer Configurations

Threat Model:

Attack Techniques Mitigated: Scanning

The evolution of configurations over time effect the lifetime of exploits and the varying configurations amongst systems helps prevent exploits from working against multiple machines.

Description:

Aims to find more secure configurations of systems over time using ideas from genetics.

Cont.

Moving Attack Surface for Web Services

Threat Model:

Attack Techniques Mitigated: Code Injection, Control Injection, and Scanning

Can help mitigate a variety of attacks. Since the service is being served randomly between systems with different frameworks, libraries, architectures, virtualization technologies, and layouts, it is more difficult to construct exploits that will work under every platform.

Description:

This technique employed diversification at different levels of a system and across many systems to create a varying attack surface across all the systems.

Cont.

Lightweight Portable Security

Threat Model:

Attack Techniques Mitigated: Code Injection and Control Injection

Helps mitigate persistent threats on a system by ensuring the OS boots into a clean and known-good state

Description:

This technique protects a user session by booting into a known good and clean state. There are two primary use cases for this technique.

Dynamic Networks

Dynamic Network Address Translation

Threat Model:

Attack Techniques Mitigated: Scanning, Resource, Spoofing, and Data Leakage

This technique assumes the hosts and entities employing this technique are safe. It can help mitigate scanning attacks by obfuscating various parts of network packet headers but not the payload of the packets.

Description:

Dynamic Network Address Translation (DYNAT) is a protocol obfuscation technique. The idea is to randomize parts of a network packet header.

Cont.

Revere

Threat Model:

Attack Techniques Mitigated: Resource, Spoofing, and Data Leakage

This technique can help protect against a couple of classes of attacks to some degree. It helps protect against resource attacks like denial of service or manipulating content on the network.

Description:

Revere is a technique that involves creating an open overlay. An overlay network is an example of a dynamic network in that it can change paths, reconfigure, and respond to links or nodes going down dynamically.

Cont.

Randomized Intrusion-Tolerant Asynchronous Services

Threat Model:

Attack Techniques Mitigated: Resource, Exploitation of Privilege/Trust, Scanning

This technique is meant to impede an attacker from manipulating messages on the network or taking a service offline.

Description:

Randomized Intrusion-Tolerant Asynchronous Services (RITAS) is a technique that builds a set of fault-tolerant consensus-based protocols on top of TCP and the IPSec protocol.

Cont.

Network Address Space Randomization

Threat Model:

Attack Techniques Mitigated: Resource and Scanning

This technique was designed to mitigate and slow the effects of an IP address hitlist-based worm.

Description:

Network Address Space Randomization (NASR) is a technique that involves changing the IP address of systems more frequently.

Cont.

Mutable Networks

Threat Model:

Attack Techniques Mitigated: Resource and Scanning

The shifting IP addresses would make it more difficult for an attacker launching denial of service type attacks against individual systems in the network.

Description:

A Mutable Network (MUTE) is a technique that involves changing IP addresses, port numbers, and routes to destinations inside of a network.