Survey of Cyber Moving Targets

Presented By Sharani Sankaran

Moving Target Defense

- A cyber moving target technique refers to any technique that attempts to defend a system and increase the complexity of cyber attacks by making the system less homogeneous,less static, and less deterministic
- It mainly aims to substantially increase the cost of attacks by deploying and operating networks/systems to makes them less deterministic, less homogeneous, and less static.
- They continually shift and change overtime to increase complexity and cost for attackers, limit the exposure of vulnerabilities and opportunities for attack, and increase system resiliency.
- They are altered in many ways that are manageable by the defender yet make the attack space appear unpredictable to the attacker.

- **Dynamic Runtime Environment:** Techniques that change the environment presented to an application by the operating system (OS) during execution dynamically.
- **Address Space Randomization:** Techniques that change the layout of memory dynamically.This can include the location of program code, libraries, stack/heap, and individual functions.
- **Instruction Set Randomization:** Techniques that change the interface presented to an application by the OS dynamically. The interface can include the processor and system calls used to manipulate the input/output (I/O) devices.
- **Dynamic Software:** Techniques that change application’s code dynamically. The change can include modifying the program instructions, their order, their grouping, and their format.
- **Dynamic Data:** Techniques that change the format, syntax, encoding, or representation of application data dynamically.
- **Dynamic Platforms:** Techniques that change platform properties (e.g., central processing unit

Cyber Kill Chain

- **Reconnaissance:** The attacker collects useful information about the target.
- **Access:** The attacker tries to connect or communicate with the target to identify its properties (versions, vulnerabilities, configurations, etc.).
- **Exploit Development:** The attacker develops an exploit for a vulnerability in the system in order to gain a foothold or escalate his privilege.
- **Attack Launch:** The attacker delivers the exploit to the target. This can be through a network connection, using phishing-like attacks, or using a more sophisticated supply chain or gap jumping attack (e.g., infected USB drive).
- **Persistence:** The attacker installs additional backdoors or access channels to keep his persistence access to the system.

Threat Model

- Data leakage attacks, e.g., steal crypto keys from memory
- Denial of Service attacks, i.e., exhaust or manipulate resources in the systems
- Injection attacks
- Code injection: buffer overflow, ROP, SQL injection
- Control injection: return-oriented programming (ROP)
- Spoofing attack, e.g., man-in-the-middle
- Authentication exploitation: cross-cite scripting (XSS)
- Scanning, e.g., port scanning
- Physical attack: malicious processor

DYNAMIC RUNTIME ENVIRONMENT- ADDRESS SPACE RANDOMIZATION

- Threat Model: Code Injection and Control Injection.
- This technique defends against buffer overflow attacks on the stack and heap from an adversary that can provide arbitrary input to a vulnerable program.
- A buffer overflow attack occurs when an attacker can provide malformed input to a program that causes it to write the input incorrectly to areas outside the allotted memory location.
- This technique performs stack randomization at both the user and kernel levels. Userlevel permutation includes both a coarse randomization (code and data segments are randomly placed) and a fine-grained randomization (functions and variables are randomized inside code and data segments).
- All programs running on the machine are protected from code or control injection through individual, independent program randomization.
- This technique could be deployed on any generic machine.
- Kill Chain Phases: Exploit Development,Attack Launch.
- Memory randomization is more effective when it is combined with various types of memory guards

INSTRUCTION SET RANDOMIZATION- GFree

- Defense Category: Dynamic Runtime Environment.
- Threat Model: Control Injection.
- This technique aims to mitigate ROP attacks against executables compiled with the modified compiler.
- ROP attacks consist of an attacker redirecting control of a program back into itself at specific useful sequences of instructions.
- This technique protects all binaries compiled with the modified compiler.
- It can be deployed on any generic machine by modifying the compiler.
- Kill Chain Phases: Exploit Development, Attack Launch.
- The encryption used is simply XOR so this technique relies on the fact that the attacker cannot read portions of the memory.
- An OS-level protection against ROP is necessary to defend against ROP in all the libraries and applications.

DYNAMIC SOFTWARE- SOFTWARE DIVERSITY USING DISTRIBUTED COLORING ALGORITHMS

- Threat Model: Code Injection.
- This technique reduces the number of machines an attacker can successfully compromise in a network using code injection attacks.
- The overall network is protected from easy compromise by an attacker.
- The approximation algorithm used for assigning versions is distributed meaning that it must be run on every computer in the network. It could also be deployed from a centralized server that is distributing software to the network.
- Kill Chain Phases: Exploit Development, Attack Launch.
- This technique relies on already having diversified versions of the applications available.
- The proposed idea is more a planning tool than a stand-alone technique. Also even assuming that diversity can stop large-scale attacks, this method does not stop attacks against one machine .
- The actual impact of diversity on successful attacks must be studied and analyzed

PROACTIVE OBFUSCATION

- Defense Category: Dynamic Software.
- Threat Model: Code Injection and Control Injection.
- This technique aims to mitigate buffer overflows and other injection attacks on network visible services.
- It mainly protects servers.
- It can be deployed on any server with important trusted services.
- Kill Chain Phases: Exploit Development, Attack Launch.
- This method does not propose a new randomization technique and relies on existing diversification techniques.
- This technique does not protect against information leakage that happens on one replica.
- This technique ensures correct responses by voting amongst the replicas, but it does not ensure that individual replicas cannot cause damage locally

Dynamic Networks- DYNAMIC NETWORK ADDRESS TRANSLATION

- Threat Model: : 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.
- Dynamic Network Address Translation (DYNAT) is a protocol obfuscation technique. The idea is to randomize parts of a network packet header. This randomization can make it more difficult to determine what is happening on a network, who is communicating with whom, what services are being used.
- This technique aims to protect the network traffic as it is traveling between systems.
- It can be deployed to workstations, servers, routers,and gateways. This could be used to protect switched local area network (LAN) segments, contention-based LAN segments, LAN-to-LAN connections
- Kill Chain Phases: Reconnaissance, Access.
- This technique does not do anything to change packet sizes, vary packet timing, or use dummy packets so it is susceptible to traffic analysis. More importantly, this technique only limits reachability

Dynamic Platforms- Security Agile Tool-Kit

- Defense Category: Dynamic Platforms.
- Threat Model : Exploitation Of Trust.
- It allows the injection of greater access control mechanisms with the ability to change them during program runtime.
- This technique protects the OS when suspicious activity or threats are detected.
- It is mainly implemented in the OS at kernel level.
- The idea is that if a detection of a certain threat or activity is encountered, the dynamic security policy of the affected applications can be dynamically changed.
- Kill Chain Phases: Exploit development , Persistence.
- An attacker could also potentially use the policies to cause a denial of service to the system by intentionally triggering the strict policies.

REVERE

- Threat Model: 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. The effects of denial of service attacks are mitigated by the distributed and well connected nature of the overlay network.
- This technique protects the integrity and availability of content delivered over a network.
- This technique would be deployed as a client on a system that wishes to participate in the overlay.
- Kill Chain Phases: Reconnaissance,Access.
- there are backup private keys available to use if one is compromised but, if an 64 attacker is able to compromise one, it is not unreasonable to conclude he could compromise the backups as well. This would allow the attacker to masquerade as the distribution center and push out fake updates,pollute the update repositories, or do other tampering of the content.
- : A dynamic network solution combined with other host protection techniques must be explored.

GENESIS

- Defense Category: Dynamic Platforms.
- Threat Model : Code Injection and Control Injection.
- This technique involves applying runtime software transformations to a program. The program is run in an application-level VM called Strata. Strata with software dynamic translation can change a program by injecting new code, modifying existing code, or controlling program execution in some manner.
- This technique helps protect the OS by making applications more difficult to exploit.

N-VARIANT SYSTEMS

- Defense Category: Dynamic Platforms.
- Threat Model: Code Injection and Control Injection.
- This technique can be implemented with different application variants to target specific threats.
- 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.
- The idea behind this technique is to run multiple variants of the same application simultaneously without relying on anything to be secret. It contains a polygrapher, the application variants, and the monitor.
- This technique protects the OS by making the exploitation of an application more difficult
- This technique is built into the OS and wrappers are created for some system calls.
- Kill Chain Phases: Exploit Development.
- This technique does not protect against application-level attacks. Also, the monitor can be a single point of failure
- This technique could also be enhanced to work with more system calls and OS components like other similar techniques. The impact of diversification techniques on attacks must also be studied.

TRUSTED DYNAMIC LOGICAL HETEROGENEITY SYSTEM

- Defense Category: Dynamic Platforms.
- Threat Model: Code Injection, Control Injection, Scanning, and Supply Chain.
- The Trusted dynAmic Logical hEterogeNeity sysTem (TALENT) is a technique that involves making a running application migrate between different platforms.
- This technique can help mitigate a OS and architecture dependent attacks.
- Since the application is migrating between systems with different libraries it is more difficult to construct exploits that will work under every platform
- This technique protects the OS and applications running on it by continually shifting the attack surface.
- This technique relies on a detection mechanism for effective jumping if it is not using a random jumping scheme.
- The technique does not provide any protection against higher-level protocol attacks. It also does not protect against attacks on one machine.
- Kill Chain Phases: Reconnaissance,Exploit Development

INTRUSION TOLERANCE FOR MISSION-CRITICAL SERVICES

- Defense Category: : Dynamic Platforms.
- Threat Model: Resource such as DoS and data integrity attacks.
- This technique aims to make critical web services more survivable in the face of attack.
- This technique protects specific applications and services to ensure they continue to operate under attack.
- Kill Chain Phases: Exploit Development, Persistence.
- This technique assumes that only one active platform will be compromised at any given execution cycle.
- This technique does not stop any attack. It just tries to mitigate DoS attacks by resource management.

GENERIC INTRUSION-TOLERANT ARCHITECTURES FOR WEB SERVERS

- Defense Category: Dynamic Platforms.
- Threat Model: Code Injection, Control Injection, and Scanning.
- Each request can use a different subset of diversified servers and the results are voted on making it more difficult for one attack to be universal.
- The first is an agreement protocol. This is a voting mechanism to determine if a server was corrupted.
- The next detection mechanism used is a combination of network and host-based intrusion detection system. Host-based intrusion detection systems are placed on every host and a network based intrusion system is used to monitor the traffic between servers and proxies.
- This technique is used to protect the availability and integrity of web services.

SELF-CLEANSING INTRUSION TOLERANCE

- Defense Category: Dynamic Platforms
- Threat Model: Code Injection and Control Injection.
- The self-cleansing intrusion tolerance (SCIT) technique aims to decrease the exposure time of a system by rotating it with copies.
- This technique protects servers by limiting their exposure time.
- If the attacker can find a working exploit against one system, it would work on all systems at once. In fact, since exploits work very fast, this technique provides little protection.
- It may also be possible to combine this technique with other movement methods on the web servers.

GENETIC ALGORITHMS FOR COMPUTER CONFIGURATIONS

- Defense Category: Dynamic Platforms.
- Threat Model: Scanning.
- This techniques aims to protect the OS or servers by finding better configurations over time.
- A stealthy attacker could still carry out their attack. It may also be possible to manipulate the configuration selection by causing detections on configurations the attacker does not want.
- A possible future direction for this project would be to expand it from simple configurations to actual software as well. It might try running a service with a specific server then later try running the same service with a different server that provides similar functionality.

LIGHTWEIGHT PORTABLE SECURITY

- Defense Category: Dynamic Platforms.
- Threat Model: Scanning, Code Injection, Control Injection.
- The primary function of this technique is to protect a web service, but the diversification also helps protect the OS as a whole
- This technique employed diversification at different levels of a system and across many systems to create a varying attack surface across all the systems.
- Diversification is done at the web server level by choosing different open source and commercial web servers.
- This diversification is done at the application level by using different implementations of the web framework

- Kill Chain Phases: Reconnaissance , Exploit development and persistence.

- This technique does not protect against web service logic bugs or failure to sanitize input. Attacks like SQL injections could be leveraged if the service does not properly sanitize input or put other mitigations in place.
- This technique does not protect against data leakage attacks or attacks against
- This technique could potentially be combined with additional internal OS movement techniques to slow down the attackers further giving the system additional time to migrate system’s one machine.

DATA DIVERSITY THROUGH FAULT TOLERANCE

- Defense Category: Dynamic Data
- Threat Model: Resource.
- This technique aims to increase the fault tolerance of an application by reevaluating the input to a program using a different algorithm. These different algorithms can produce exact equivalent output .
- The technique builds on the idea of N-version programming but uses a data-centric version.
- This technique aims to protect a program by ensuring the output is acceptable.
- Kill Chain Phases: Exploit development.
- If an attacker is attempting to corrupt or manipulate the output of a program, this could make it more difficult. The technique is mainly focused on integrity protection.
- There are many applications and services now that have very dynamic and varying outputs so it may not be trivial to determine if output is valid

REDUNDANT DATA DIVERSITY

- **Defense Category:** Dynamic data.
- **Threat Model :** Resources and Code Injection.
- **Kill Chain** Phases**:** Exploit development.
- This technique is a variation of the N-variant programming technique which involves running multiple copies of a program that each run transformations of the original data being protected without having to rely on secrets.
- This was implemented to protect user identification (UID) and group identification (GID) that are used for determining permissions.
- This technique was currently only implemented to protect data inside of the application
- This technique aims to protect data entities inside of a running program on systems.

- An attacker could still target data parts of an application that are not randomized if they can be used to mount an attack.
- It could also use advanced control injection attacks that could still potentially affect many or all variants

Data Randomization

- **Defense Category :** Dynamic data.
- **Threat Model :** Code Injection and Control Injection.
- This is a compiler-based technique that provides probabilistic protection by randomizing all the data that it stores in memory.
- This technique protects the data applications store in memory.
- **Kill Chain Phases:** Exploit development
- An attacker would have to find a method to either leak the keys or guess the keys used to randomize the data.
- This technique would be to combine with other memory protection techniques such as address space randomization.

End To End Software Diversification

- Defense Categories : Dynamic data , Dynamic Software.
- Threat Model: Code Injection , Exploitation of Authentication.
- Design: The main idea 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.
- The proposed diversification methods include changing HTTP,HTML,DOM, web server instruction set/memory layout, and local files used by the servers .
- Each user instance has a different randomization plan. It could also be implemented that each user request causes a new randomization of some of the methods
- This technique mainly aims to protect web server.
- Kill Chain Phases: Exploit development ,Attack launch

THANK YOU