Cyber Defense: A Fighting Chance

A research proposal by SPC Todd (17C)

9/26/2021

B CO, Cyber Training Battalion, Ft. Gordon

Security Note: The following report is not informed by any official knowledge of nation-state technology and contains no classified information. It is based only on logical conclusions and open-source research.

Executive Summary

The goal of this research is to guarantee the defender a fighting chance by challenging the three most critical cyber security problems with fundamental solutions.

The first problem challenged by this research is one of **visibility**. An unseen threat is difficult to defeat. Between two kernel processes battling at ring zero, the defender fundamentally loses. Even a theoretical perfect scanning system fundamentally still loses a cat-and-mouse game where the defensive tool must be published by necessity for widespread and long-term use while the attacker can observe and adapt secretly. The win condition is the attacker locating the result of said scan in memory and overwriting it. Or the attacker can simply turn the scan off. This problem cannot be truly circumvented within the computer system. There are no endpoint security logs to analyze when a sufficiently sophisticated attacker operates from kernel mode.

A wide array of equipment is vulnerable: routers, firewalls, IDS, workstations, VM clients, the hypervisor that oversees them[40], the domain controller, *everything* is vulnerable. Even systems thought to be the few bastions of security such as the UEFI boot process with Intel Boot Guard enabled have been demonstrated to have such vulnerabilities. The very processor mode which enforces these protections[36, 37] has been demonstrated to be vulnerable[35]. It is only a matter of time before this happens again if such an exploit is not already in use today. This payload might operate with kernel mode privilege, effectively invisible to internal scanning for the reasons laid out previously. The defender, whose role it is to detect and mitigate this threat's destructive effect, cannot effectively perform this role without being able to see the target.

The second problem challenged by this research is one of **recognition**. Being able to see the threat is not enough if it cannot be recognized for what it is. Just like the same idea might be communicated through infinite combinations of words, a programmed effect can be expressed through infinite combinations of machine code instructions. Human analysts can recognize an obfuscated program for what it is, given time, but when a disruptive effect can be executed in a few microseconds time is limited. Computer programs designed to achieve this analysis goal in real-time have historically proven ineffective. Adversaries use increasingly complex strategies to outsmart programs designed for recognition:

- State-of-the-art obfuscation defeats most signaturing attempts outside of behavior analysis.
- Evasive strategies evade sand-boxing[2, 41] effectively, making behavior analysis difficult.

- Return Oriented Programming (ROP) jumps through benign code to perform malicious goals[3] and defeats the common Write or Execute (W^X) defense[10].
- Ring 0 exploitation allows for simply turning off a defensive scan or overwriting scan results and hiding from logging systems.

The third problem challenged by this research is one of rapid **recovery**. The best-case scenario for detection of a zero-day exploit executing a novel payload is prevention of re-execution after initial detection and signature development. A rapid recovery solution is critical. Recovery delays in private industry can cause damages in the billions, but for the military the losses accrued might be priceless lives. The problem of recovery is not one rooted in some sophisticated technical challenge but simply involves time, cost, and human reliability. Backup-storage disconnected from any network (cold storage) is all that is needed to recover data after a disruptive cyber event. This is easier said than done. It requires a physical task to be executed, usually by a human worker, correctly on a regular basis. Data storage is also expensive, but even more expensive is duplicate storage. When expense is not a problem, access latency on cold storage can be. Reconnecting the disconnected storage to recover onto live systems can take days. Days of additional disruption for cold-storage access can be a costly outcome after an already disruptive attack. As a result, cold storage is often neglected and when it is not, latency of retrieval leads to extended disruption anyway.

These problems of visibility, recognition, and recovery can be overcome. For two reasons:

1) There exists a level of privilege even more absolute than kernel mode. Not System Management Mode (SMM), sometimes referred to as ring –2[15]; even more absolute than that. Direct physical hardware access is available to the defender to guarantee unfettered visibility into the system. The JTAG protocol implemented on virtually every computing device allows anyone with physical access to a machine absolute control to execute instructions on each physical core of any CPU at a limited rate. Most industrial hardware of interest in this field also has detachable RAM which can be supplemented with a memory injection capability. In addition to visibility assurance, hardware access grants the defender the ability to ensure integrity of some other useful functions, which will be explained. This can be achieved with the right tools.

2) Obfuscation can be defeated. There is no question as to whether this is possible, human analysts already do it on a regular basis. It is only a matter of finding a rapid automated approach robust

enough to fundamentally prevent evasion. As a result, source-code engineered by an attacker can be rendered useless. Military-grade cyber toolsets are neither cheap nor fast to build. Any given nation can only have so many people with the security clearance to work on these projects. Therefore, if one defeats obfuscation and burns the source-code of these toolsets it is a severe problem for the adversary logistically.

A three-component solution is proposed to leverage these two facts in order to improve mitigation of kernel level attack vectors. This is proposed though defeating obfuscation, ensuring data integrity, and ensuring the integrity of a kernel-level scanning capability.

- **Payload identification:** A scanning technique which defeats state-of-the-art obfuscation in order to burn a payload's source code without needing access to that source code – only an obfuscated version of it. This technique is borrowed from recent PhD research which treats small pieces of a program as individual black boxes and solves their functionality via the Monte Carlo tree search. This is combined with adversarial forced branch emulation. The researchers combine these two approaches to find the simplest representation of a section of code and forcibly analyze hidden behavior. Obfuscation, emulator evasion, and Return Oriented Programming are defeated with a high rate of success using this strategy. As originally presented this approach was not fast enough for live scanning but some innovations solve this problem. Just-in-time partial scanning of the program's immediate future is proposed in place of a start-to-finish branch exploration process.

- **Scan integrity assurance:** The best scanning technique imagined is utterly ineffective against a kernel-mode attacker who can just turn it off or overwrite its results in memory. External scanning through direct memory access and an external CPU would be an ideal solution, however it would also be extremely expensive for obvious reasons. Instead, for scan integrity assurance, a hardware-enforced memory injection functionality is integrated into the PCB of a custom DIMM to physically inject the scanning software into secret locations in memory. JTAG is leveraged to execute that injected code with kernel-mode privilege. The injected memory range is replaced by original contents after each scan and a new location physically injected for the next scan. In this way, it is statistically impossible for the attacker to find the scanning program in time to interfere.

- **Data integrity assurance:** Prevention of payload re-execution is not helpful if the attacker was able to perform data-destruction upon the first execution. A hardware-enforced storage solution is proposed which ensures an attacker cannot compromise data integrity until a pre-determined time-frame has passed. During this time the defender has a guaranteed opportunity to perform incident response, hunt the threat, and recover before protected data can be destroyed. This is achieved by a device which plugs into any standard storage connector and simulates first-in-first-out cold storage without the user having to perform the physical connection and disconnection of a backup drive.

The solutions propose to defend numerous critical hardware devices through plug-and-play installation. Support includes any computing device with detachable RAM and storage:

- Workstations
- Servers
- Routers
- Switches
- Firewall, IDS, IPS systems

The toolset proposed leverages plug-and-play hardware and takes steps to minimize the necessary computation performed externally on the micro-controllers involved (lowering required cost). The tools are sophisticated in nature, but trivial to install and not materially expensive. In other words, the cost of research and development will be high, but the material cost of manufacture is low, and this should translate to a low production cost so long as a large quantity of the toolset is manufactured. Thanks to the modular plug-and-play nature of the proposal, cost of deployment and maintenance will be as low as any hardware component installation can be, although still a considerable expense compared to software-based solutions which can be installed automatically but fail to effectively solve this problem.

The solutions described will provide leaders with new ways to accomplish the defensive cyberspace mission. These proposals are too broad to be thoroughly explored by any single researcher and this paper does not attempt to do so. The proposal here is only an initial look at the solutions, and a pitch to leaders for resources to be allocated for a team effort to be put toward developing more thorough specifications on the subject.

Leveraging the Physical Access Advantage

Without doubt, a purely software-based solution would be far cheaper to employ. However, such a defense will never be adequate. At the maximum privilege levels achieved by nation-state level attackers, a defending program cannot fundamentally outmaneuver an attacking one. It is the perfect cat-and-mouse game. Each side of the battle has infinite ways to counter its opponent. However: Because the defense program must be published in order to be widely used, it has potentially already been reverse-engineered and the attacking program modified to subvert it. An attacking program can be kept in secret until the moment of attack. This fundamental difference means that the defending program will always be at a fundamental and critical disadvantage when facing an opponent with nation-state resources at its disposal.

The nation-state tier attack this report seeks to mitigate will almost certainly involve code running with the highest privilege, potentially to include ring "-2" [15] (System Management Mode) and even ring "-3" (Intel Management Engine, or Platform Security Processor) which was also demonstrated to be exploitable[31]. This should serve as a warning to anyone who considers the idea of introducing remote management of the external hardware solutions presented in this paper to very carefully limit, validate, and secure the remote update mechanism.

The defender has access to something more powerful than kernel mode: External hardware access. This is possible through engineering essentially man-in-the-middle attacks against one's own hardware through the design of components designed to intercept memory on the RAM stick and access the JTAG interface through the designated port present on virtually every computing device, or over USB. It is expensive to research, develop, and deploy physical, external security enforcement solutions. However, there it is the only way to deploy a true defensive capability against a nation-state attacker. Leveraging this advantage allows the defender a rare assurance of integrity, assuming the external hardware is properly isolated, and input validated. "Properly validated" can be a vague target in practice, which is why is it critical that the input validation circuits be verified. This can be accomplished by mathematically proving the circuits are only capable of behaving within the expected outcome range[4, 30].

The experienced systems administrator might reasonably have concerns about potential difficulties in updating the firmware of the external hardware devices described in this paper which by their very nature cannot trust the host machine they reside on. However, this can be overcome through a Diffie-Hellman key exchange, allowing the external hardware to trust a signed firmware update delivered over an untrusted host machine. A robust solution likely involves a private key locked in a safe in the CISO's office. The key must remain completely isolated from compromise and only be used to sign firmware updates, which can then be pushed out over the network and consumed by the external monitoring toolset which will externally perform validation of the certificate.

A Brief History of Malware Fingerprinting

In the 1950's researchers were just beginning to postulate theories about virulent computer programs. Previously having served as a brilliant researcher for the Department of Defense during the Manhattan Project, John von Neumann famously wrote his final published work[24, 25] on the subject – a first look at a concept which would come to be known as malware. His longtime collaborator Stan Ulam would eventually be named as a source. These two names are mentioned here not for the sake of historical interest, but because only a decade earlier Ulam and von Neumann invented the very same Monte Carlo method this paper presents as a key component in a desperately needed robust solution to obfuscation.

In the earliest days of computer malware, one could simply hash a malicious program's data and store the value in a database to be queried later. Malware soon adapted through the introduction of obfuscation. By 1986, researchers were pioneering more advanced research into the topic and exploring encryption of computer viruses[26, 27]. In 1987, the first anti-virus programs were developed [28, 29]. Antivirus companies would go on to leverage virtual machines and emulators to elicit behavior signatures from the logic in an attempt to side-step obfuscation. A cat-and-mouse game of detection and evasion ensued.

By 2016, research was published demonstrating an effective approach to defeat modern forms of behavior analysis with near trivial effort[2, 41]. Sand-boxed environments were proven to be disadvantaged on a fundamental level[2, 41]. Antivirus (now more aptly referred to as "anti-malware") will likely remain an effective solution for defeating a vast majority of less sophisticated malware, but it is not effective against state-of-the-art obfuscation.

Anew solution format combining anti-malware with a managed, multi-layered approach has risen in popularity for enterprise malware defense: The Endpoint Detection and Response[21] (EDR) genre of defense system. These systems rely on a wider range of data collection on and outside of the host machine to develop malicious signatures, primarily by converting system call I/O traces into logs. EDR offers the benefit of communicating behavioral data to an external monitoring solution, such as a Security Information and Event Management (SIEM) system. As a result, indicators can be considered in a more dynamic manner. This combination of EDR and other sensors with a SIEM system would empower more dynamic threat detection. Although the concept of zero-trust was first described in 1994[44], it was in 2018 that SANS instructors would go on to promote a Zero-Trust approach[42, 43], and a dynamic, non-binary trust system[43] known as a "trust score"[45].

SIEM systems, Zero-Trust, trust scores, and EDR all push the modern cyber security industry toward a better defensive position, but they remain insufficient[5] when the sensors they rely on are compromised by this undetected malware. No matter how effectively the defender analyzes captured data, when endpoints are affected by attackers, especially those escalated to kernel-mode, their sensors can't be trusted to send the data necessary to accomplish the desired goal. While logs which would be generated by sensors detecting credential-reuse and other suspicious activity can simply be turned off, these analysis solutions will continue to be circumvented with ease by sufficiently resourceful attackers.

Currently, despite continued efforts by cyber security professionals to engineer better solutions, none have been published which solve the evasion problem described by the AVLeak, Blackthorne et al research[2, 41], which severely reduces the effort required to detect whether the execution environment is a detection sandbox emulator or virtual machine. With this information, the payload can behave innocently within the detection environment, then perform malicious effect once running on the true machine. Numerous papers have been published on leveraging machine learning for this purpose, but most "deep learning" models currently optimize toward shortcuts to solutions rather than robust reasoning which makes them very exploitable[32, 33] and an unreliable approach to this problem.[6] The problem of fingerprinting state-of-the-art malware lingers, unsolved.

Data Integrity Assurance

Virtualized Cold Storage Buffer

Summary

Cold, (physically disconnected) storage is logistically challenging to maintain, costly, and can take days to perform recovery, but provides a critical last layer of defense to prevent extended denial of service in the case of an attack. Every organization has certain dynamic datasets critical to recovery. Destruction of such data dramatically increases the time required to recover operationally from a severe attack, costing the organization time, money, or where services are critical, potentially lives. This paper proposes a tool serving as an alternative to cold storage which:

- emulates the process virtually, reducing logistical costs.
- maintains the desired data integrity guarantees through external hardware enforcement.
- decreases data recovery delay, further reducing disruption of service related losses.

This hybrid backup storage is “cold” in the sense that, while not physically disconnected, an attacker cannot damage the protected data for a configured time-period, giving the defender a safe window during which to detect and respond to the threat.

Problem Overview

Attackers frequently target backup storage in order to ensure payment of data ransom and often succeed, resulting in severe disruption to targets who often pay the ransom. This results in a self-feeding cycle where payment generates further incentive for cyber criminals to increase efforts towards this form of attack. On the national stage, nation-state actors have demonstrated the ability to use this strategy to delay recovery following state-sponsored cyber warfare efforts against private and government entities.

It is of vital importance for both private and government entities to pursue and employ an effective solution to disrupt this cycle. Cold storage is traditionally the advised tool to achieve this. However, the physical disconnection requires manual human work, proper routing of data, and for people to do their jobs correctly every time. Often the costs associated with data duplication and physical rotation of storage drives cause businesses to forgo it.

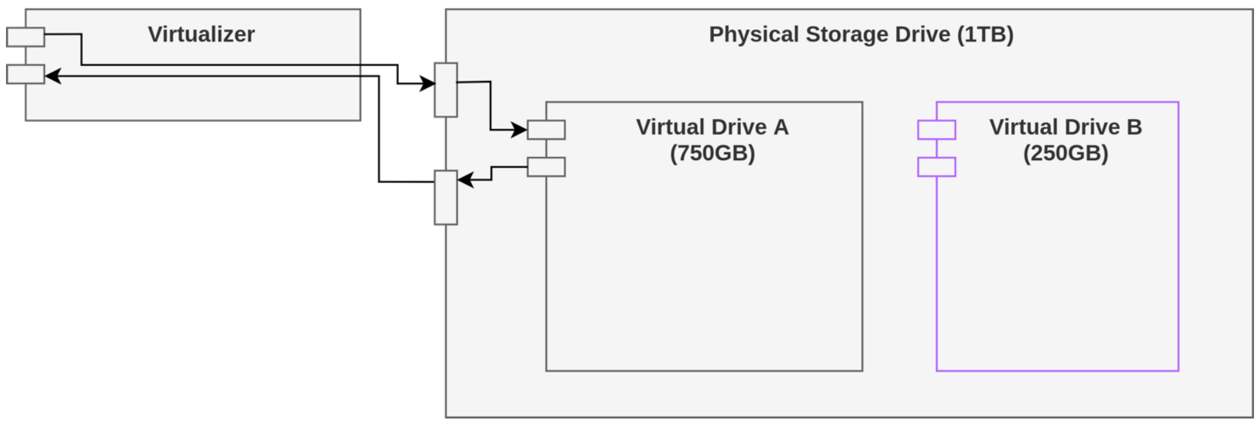
The expense that comes with duplication of data and physical disconnection is a significant factor in whether a business will choose to utilize this defensive measure. Recovery of cold storage can also take days or weeks depending on how the data is stored, as the backups must be physically located, connected, and contents distributed, potentially through a third party service.

Solution Overview

The following research proposes a solution to reduce data duplication of traditional cold storage and automate the desired outcome. The goal is to make the benefits of cold storage more accessible, preserving data integrity in an automated manner with no moving parts and less data duplication. The tool described translates to every widely used storage protocol by combining the same base technology to interchangeable modules designed to support each individual physical storage protocol (SATA, PATA, CSCI, NVMe).

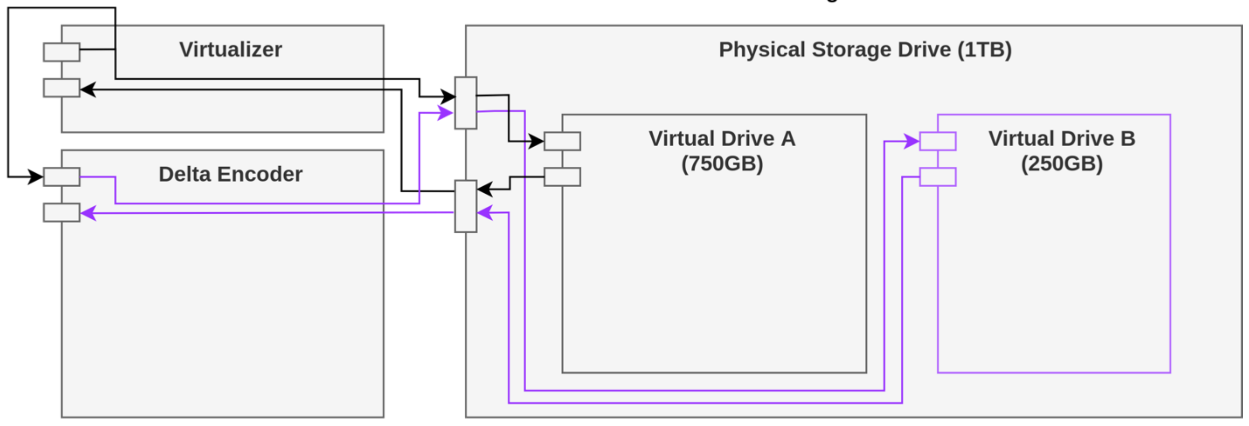
This solution proposes to replace the physical disconnection step through hardware plugged into the storage connection. It is commonly taught as a fundamental factor in cyber-security that having physical access to a device gives an attacker a potentially insurmountable advantage. Defenders always have this advantage and should leverage it. A hardware-based solution is proposed to leverage physical access to provide the data integrity benefits of cold storage without the logistical challenges presented by the standard approach.

Figure 1



1. **Figure 1:** Each storage drive connection interface is connected to the **Virtualizer**, which manipulates the input/output (I/O) scheme to make the drive *behave* as two separate physical storage drives virtually: **Virtual Drive A (VD:A)** and **Virtual Drive B (VD:B)**. VD:A interfaces with the host as a standard storage drive would. The portion allocated to each drive is administrator configured depending on the amount of data the organization needs to protect and the duration of the incident detection and response window desired.

Figure 2



2. **Figure 2:** Between the virtual interface between the Virtualizer and VD:B is a printed circuit board: **Delta Encoder.** Directories the administrator determines are necessary to protect for rapid disruption recovery are defined, that configuration is uploaded to Delta Encoder via USB port. Delta Encoder then interprets I/O events that affect a protected directory. Initially, the contents of the protected directory are cloned onto VD:B and will hereafter be referred to as the **delta base image**.

3. The Virtualizer duplicates storage I/O executed between the host and **VD:A** and passes it to the Delta Encoder. The Delta Encoder performs an XOR operation against the previous and new data states in order to generate deltas (also known as diffs), a process known as **delta encoding**. Delta encoding is a way of recording differences when files are changed, resulting in a complete history of data states while only having to store the changed bits. Addressing metadata is stored with each delta.

4. As VD:B reaches capacity, the oldest deltas are permanently applied to the delta base image. In this way, a buffer window is created during which the defender has some time range with which to notice critical destruction of data backups which would otherwise be permanently destroyed or encrypted. While an attacker may be able to corrupt some data without notice, any data that is of critical, ongoing necessity for day-to-day operations should be noticed in time to perform a recovery operation. The duration of this **recovery buffer window** depends on the amount of space allocated to VD:B and the amount of activity performed by the business.

5. In the event of an incident, the enterprise uses an application on the hosts to read the contents of VD:B and interpret the deltas to recover a desired history state within the buffer window. The circuit is designed to ensure that the only commands a host can reach VD:B with are read-only. This outcome must be supported by mathematical proof, a process in which the circuit is simple enough to calculate every possible outcome and guarantee correct results.

Persistence

The objective of this tool is to ensure operationally critical data is not permanently destroyed, corrupted, or otherwise damaged to create extended and costly disruption of operations. The attacker is still able to gain persistence in the data. This tool ensures that the defender has an opportunity to locate the threat and remove persistence from recovered data in order to rapidly recover.

Cloud Services & Virtual Machines

One might question how such a technology can be applicable to a dynamic and rapidly changing business environment which is quickly moving to cloud services and virtual machine hosts. Physically connecting to a USB port on the Delta Encoder to reconfigure which locations are protected would be logistically expensive, likely resulting is undesired directory structures wherein the administrator requires all critical files to be stored in a specific, never changing directory location. Conveniently, a cloud service is more of a benefit to the model than a hinderance. A cloud storage provider can leverage this tool to offer a more user-friendly way to change which directories are protected, while behind the scenes the actual directory stays the same and a software-feature is used to map new directory names to that same protected directory location on the drive.

Threat Model

To attack this solution, the attacker must rapidly write to the protected directory to force the history to cycle before the defender begins incident response. This much I/O in a short enough time span for this attack to succeed can be side-stepped. Fortunately, the potential false positive cost of doing so is very low and does not interrupt operational integrity. Once enough rapid I/O has occurred to reach the configured **recovery buffer window minimum**, delta encoding pauses. This event can be logged, and thresholds leading up to the event can also be logged, allowing analysis and detection of anomalous activity through a SIEM tool. I/O on the primary disk (VD:A) continues uninterrupted but the difference history (VD:B) will not receive updates. Once enough time passes to do so without violating the recovery buffer window minimum policy, the oldest delta continues being merged into the delta base image and the current state of (VD:A) is compared and entered as a series of new deltas.

As a result, the attacker can (noisily) force data difference history to halt but cannot modify the delta base image with corrupted history deltas sooner than the configured policy dictates, ensuring the defender has a predetermined timeframe to notice the data destruction. It is also important to reiterate that this system is designed to protect data critical to daily operations. Due to the nature of this system's operation, the attacker cannot damage the backups without first damaging the data's operational copy on the storage drive. In other words, it should be immediately obvious to the defender that an incident has occurred, and there is nothing the attacker can do to prevent data recovery unless the defender takes longer than the configured recovery buffer window to notice the data corruption.

Scan Integrity Assurance

Hardware-Enforced Memory Injection Defensive Scanning

Summary

This section proposes a method of leveraging hardware access to run a kernel-mode program on the CPU which cannot with consistency be detected in time by a kernel mode privileged attacker, even one which has access to the defending scan program's source code.

Problem Overview

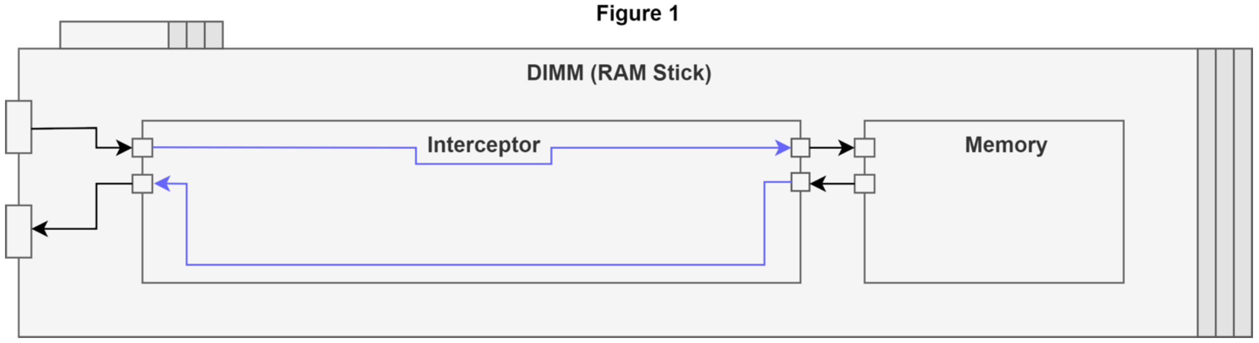
Between two rootkit processes battling at ring zero, the defender fundamentally loses. Even a theoretical, perfect scanning system which detects previously analyzed payloads with 100% success fundamentally still loses a cat-and-mouse game where the defensive tool must be published by necessity for widespread use while the attacker can adapt secretly. The win condition is the attacker simply locating the results of said scan in memory and overwriting them.

Solution Overview

To prevent an attacker from disabling or compromising the integrity of a defensive scan, an external scanning system with access to main memory is required. Rather than spend on an expensive external CPU, this proposal opts to leverage a True Random Number Generator to inject the scanning logic into an unpredictable location in unused memory. To send the instruction pointer for any desired CPU core to this injected address, JTAG is leveraged. The following diagram depicting 4 RAM sticks outlines the system described.

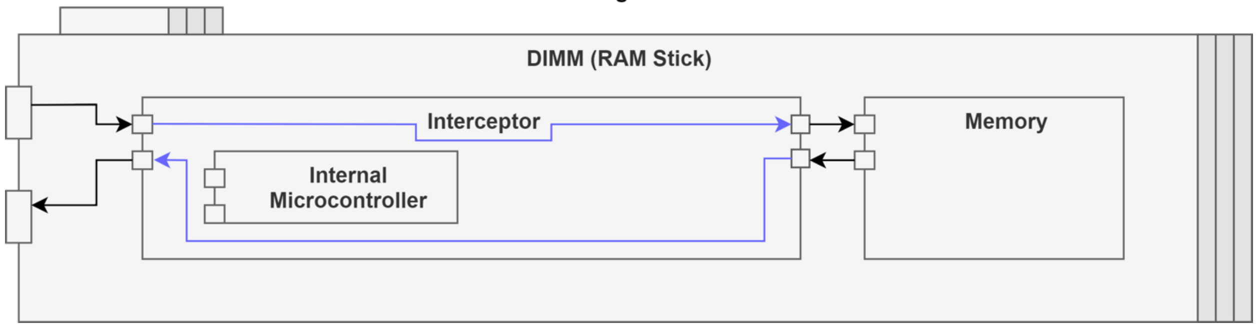
The illustrated size of the modules in the diagrams below do not represent physical size or placement and only serve to depict the relationships between systems involved.

Figure 1

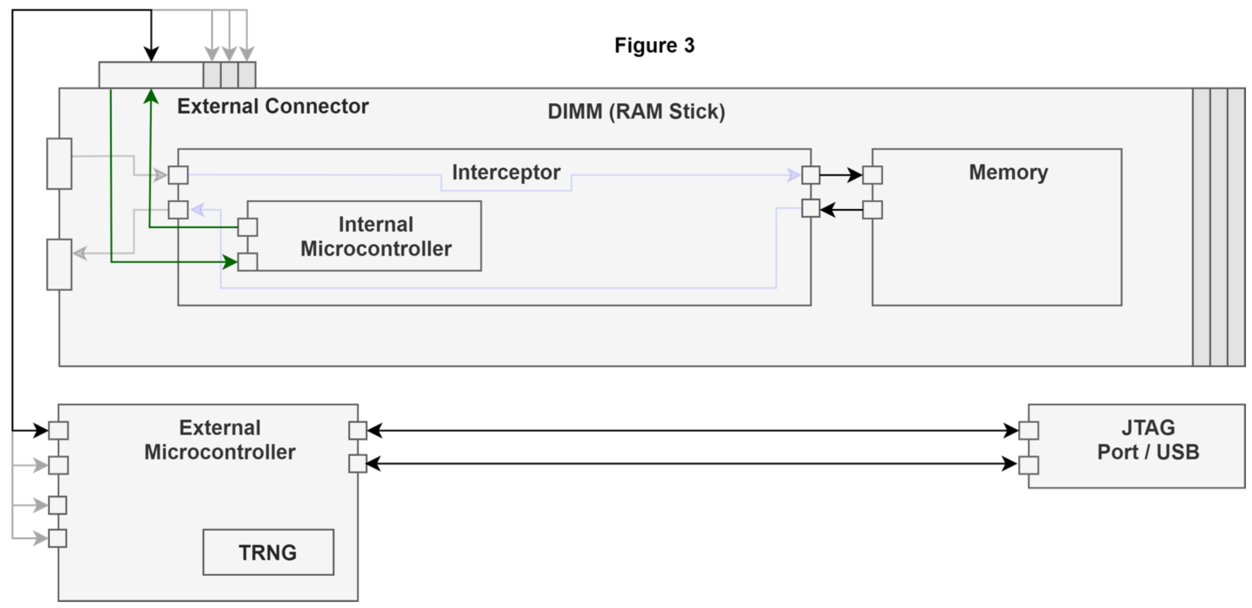


1. **Figure 1: Interceptor,** a custom micro-controller supplements the standard PCB design of each DIMM to access main memory. Write operations are uninhibited unless the external monitor is currently performing an equivalent operation to the same memory range.

Figure 2



2. **Figure 2:** An **Internal Microcontroller** built into the DIMM PCB allows memory read / write operations to be conditionally intercepted. The allows read / write access to main memory externally, outside the control of an attacker.



3. **Figure 3:** An **External Connector** positioned at the top of each DIMM (for accessibility) connects the Internal Microcontroller to a single **External Microcontroller** which acts as the main controller for the injection process. It also has access to a **True Random Number Generator (TRNG)** for secure random number generation. The External Microcontroller connects to either the JTAG port or leverages JTAG over USB in order to send the instruction pointer to the injected memory and escalate privileges.

At a set interval, the External Microcontroller executes the following process on each physical CPU core:

1. Inject a stored scanning program into an unused, random memory location. Each time memory is injected, the TRNG generates another random address which is included as a value in the injected memory. This is a secret output location which is observed by the External Microcontroller for any positive scan results. Action is then taken against the discovered attacker in the same manner.

2. Send a JTAG command to send the instruction pointer to the injected memory location and escalate privileges to kernel mode.

3. The injected memory is executed (the scan).

4. The final scan instruction returns to the previous privilege mode and sends the instruction pointer back to the original location so the interrupted thread can continue.

The injected memory range is frequently moved to a new random location for the purpose of making it statistically unlikely for an attacker to be able to locate said memory in time to interfere.

Additional Benefits

- **Payload Capture:** In addition to the benefit of scan integrity assurance, direct external memory access offers an important secondary capability: In-memory payload capture. A sophisticated attacker with kernel privilege can prevent any attempts made by the defender internally to capture the memory range it operates from. Physical, external memory access allows an effectively invisible attacker, once identified, to be observed quietly until its payload is sufficiently exposed.

- **Assured data reporting integrity to SIEM:** Thanks to ensured kernel-mode execution integrity, the scanner can also use a network driver of guaranteed integrity to report scan data over the network without risk of being disrupted by a maliciously modified driver.

Threat Model

The only route for an attacker to defeat the injection mechanism described is related to the requirement for performance. Depending on how often the scan is triggered, and how effectively one can attach to newly executing kernel code to ensure scan coverage, the attacker's most promising option might involve attempting to operate for only small durations of time or evading the scan's obfuscation-defeat capability. It is unclear whether either of these possibilities are plausible weaknesses, but every effort should be made during further iteration and testing of this capability to attempt to exploit them and pivot the approach if necessary. The external-hardware approach places the advantage on the defender's side, but that does not guarantee success if the threat model is not carefully considered and well-tested against.

Obfuscation Defeating Payload Identification

A Monte Carlo Tree Search, Black-box Approach to Obfuscation

Summary

Program Synthesis, a technique invented by PhD students at Germany's Ruhr University, presents a groundbreaking approach to defeat state-of-the art obfuscation. The reader is encouraged, if mathematically inclined, to consider reading the full research paper: Syntia: Breaking State-of-the-Art Binary Code Obfuscation via Program Synthesis. Blazytko et al presents a potential leap forward in the problem-space of obfuscation. The research will only be summarized here. There is also a video presentation, at Black Hat 2018 for any reader interested less in the mathematical concepts and more in a practical explanation.

It involves defeating obfuscation by treating small code chunks as black-boxes and using the Monte Carlo tree search to find the simplest representation of the logic. It is a powerful approach and could be the perfect foundation for a state-of-the art fingerprinting system. The Monte Carlo algorithm was invented for the Manhattan Project by researchers Stanislaw Ulam and John von Neumann. It translates extremely well to this problem-set.

Much like one can express the same idea through an effectively infinite number of word combinations, functions in code can be obfuscated to take any number of randomized routes to express a desired logical effect. Just as a human will recognize an idea expressed through many different phrases, a computer can assess an obfuscated version of a specific source-code function and recognize the core logical effect being expressed. Programs are just a series of inputs converted to outputs through a very specific conversion function and no matter how that conversion is obfuscated, the same conversion is still there and can be reduced. Analysts already do this manually in reverse engineering. The only challenge left in order to burn the adversary's source code is to automate that process. The technique presented by Blazytko et al is demonstrated to achieve this goal.

Effectiveness

Blazytko et al discovered that by deriving sequentially executed instruction traces from forces branching outcomes, treating those sequences as black-box samples, and then applying a Monte Carlo tree search algorithm to efficiently discover a solution, complex code obfuscation could be simplified down to simple patterns, regardless of evasive behavior. The researchers did not follow every branch or test every code segment but forced conditional branching paths and samples regardless of conditional value. In addition to state-of-the-art obfuscation, this approach proved effective against obfuscation achieved through Return Oriented Programming (ROP) gadgets.

The Monte Carlo Method

"The Monte Carlo method, which uses random sampling for deterministic problems which are difficult or impossible to solve using other approaches, dates back to the 1940s." - summary by *Wikipedia*, source: Metropolis, Los Alamos Science Special Issue 1987

The modern version of the approach was developed by U.S. Government researchers to solve for unknown functions within physics equations pertaining to the Manhattan Project.

"In the late 1940s, Stanislaw Ulam invented the modern version of the Markov Chain Monte Carlo method while he was working on nuclear weapons projects at the Los Alamos National Laboratory. Immediately after Ulam's breakthrough, John von Neumann understood its importance. […] Despite having most of the necessary data, […], the Los Alamos physicists were unable to solve the problem using conventional, deterministic mathematical methods. Ulam proposed using random experiments." - summary by *Wikipedia*, source: Stan Ulam, John von Neumann, and the Monte Carlo Method, Roger Eckhardt, Los Alamos Science Special Issue 1987

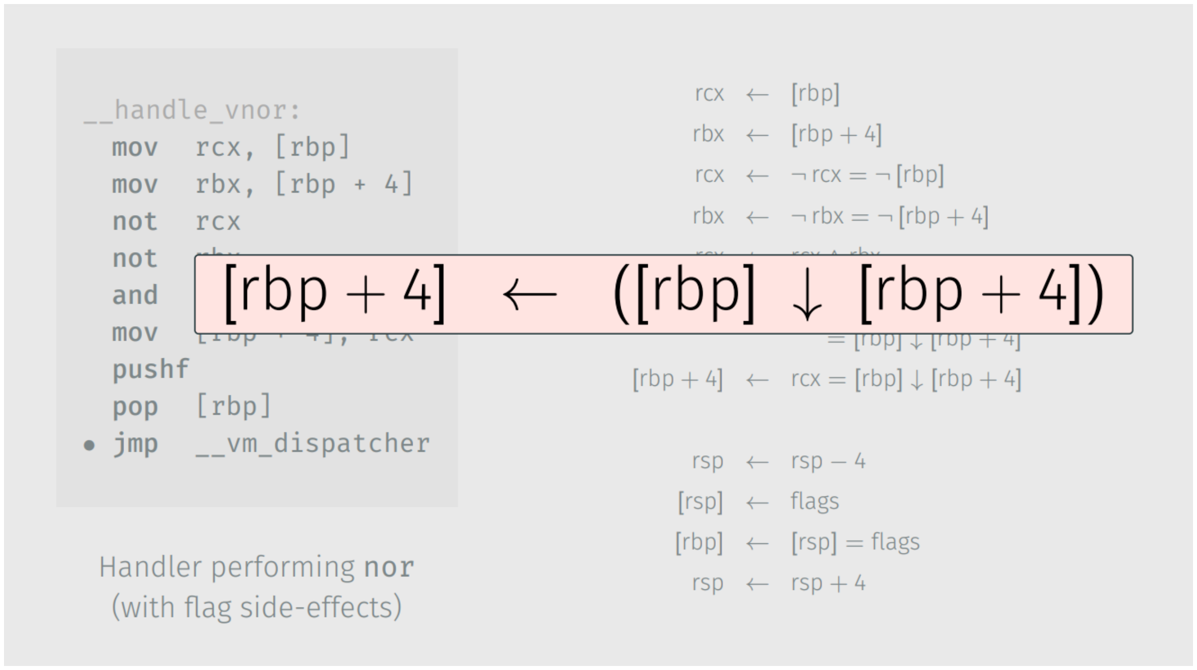
The method has a history of successful usage to solve for unknown functions throughout the past century.

"In his 1987 PhD thesis, Bruce Abramson combined minimax search with an expected-outcome model based on random game playouts to the end, instead of the usual static evaluation function. Abramson said the expected-outcome model 'is shown to be precise, accurate, easily estimable, efficiently calculable, and domain-independent.'" - summary by *Wikipedia*, source: Abramson, The Expected-Outcome Model of Two-Player Games, Columbia University, 1987

In short, this approach, invented to solve for unknown functions, is now exactly what is needed to solve for the reduction of obfuscated instruction traces. Blazytko et al illustrate the approach aptly:

$$\begin{tikzcd}
 & & & U \arrow[dll] \arrow[dl] \arrow[dr] \arrow[drr] & & \\
 & U\ U\ * & U\ U\ + \arrow[dll] \arrow[dl] \arrow[dr] \arrow[drr] & & a & b \\
U\ b\ + & U\ U\ U\ +\ + & & U\ a\ + \arrow[dl] \arrow[dr] & U\ U\ U\ *\ + & \\
 & & U\ U\ *\ a\ + & & b\ a\ + &
\end{tikzcd}$$

At the lowest level, every program, script, or command on a modern computer executes assembly similar to the following illustration. An obfuscated set of instructions are presented, and simplified down to their simplest form automatically, leveraging the Monte Carlo tree search.



```
__handle_vnor:
  mov   rcx, [rbp]
  mov   rbx, [rbp + 4]
  not   rcx
  not
  and
  mov   [rbp + 4], rcx
  pushf
  pop   [rbp]
• jmp   __vm_dispatcher
```

$\mathrm{rcx} \leftarrow [\mathrm{rbp}]$

$\mathrm{rbx} \leftarrow [\mathrm{rbp}+4]$

$\mathrm{rcx} \leftarrow \neg\,\mathrm{rcx} = \neg\,[\mathrm{rbp}]$

$\mathrm{rbx} \leftarrow \neg\,\mathrm{rbx} = \neg\,[\mathrm{rbp}+4]$

$= [\mathrm{rbp}] \downarrow [\mathrm{rbp}+4]$

$[\mathrm{rbp}+4] \leftarrow \mathrm{rcx} = [\mathrm{rbp}] \downarrow [\mathrm{rbp}+4]$

$\mathrm{rsp} \leftarrow \mathrm{rsp} - 4$

$[\mathrm{rsp}] \leftarrow \mathrm{flags}$

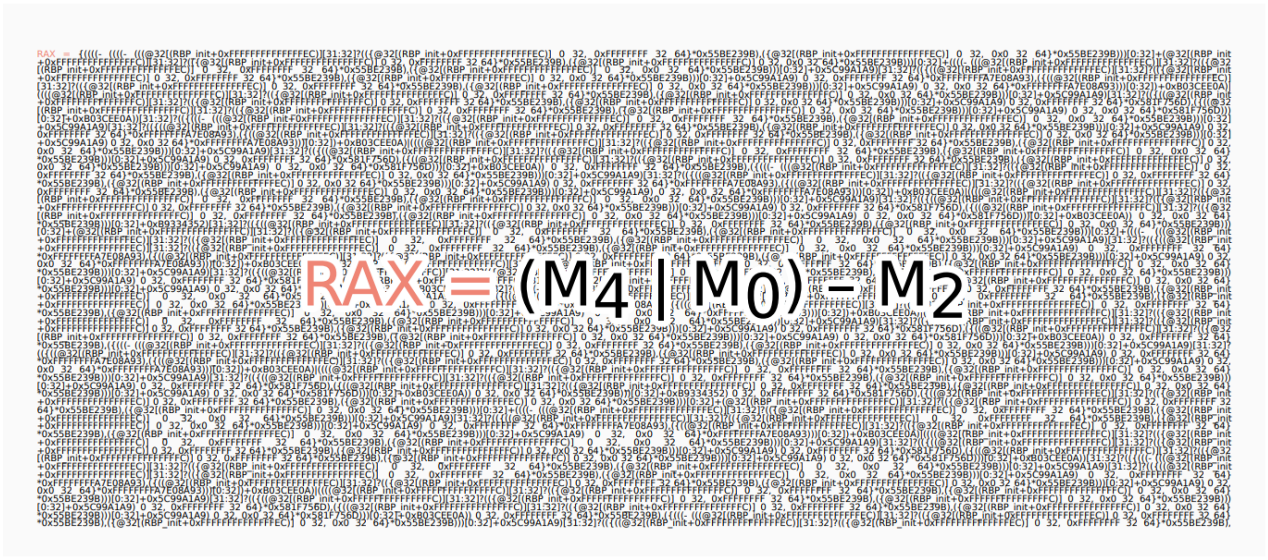
$[\mathrm{rbp}] \leftarrow [\mathrm{rsp}] = \mathrm{flags}$

$\mathrm{rsp} \leftarrow \mathrm{rsp} + 4$

$$[\mathrm{rbp}+4] \leftarrow ([\mathrm{rbp}] \downarrow [\mathrm{rbp}+4])$$

Handler performing nor
(with flag side-effects)

Even an instruction trace as complex as the noise printed below can be simplified to a small sequence through this method, referred to by Blazytko et al as Program Synthesis.



The Institute for Defense Analyses would be an excellent place, perhaps the best in the world, for interested leaders to request expert analysis of the potential this approach has for aiding in mission success. In fact, in would likely be appropriate for the Insititute to be heavily involved in the assessment and engineering of this deeply mathematical proposal.

Of critical importance is the detail that the researchers successfully applied it to Return Oriented Programming (ROP) based obfuscation[3]. This is a particularly difficult form of obfuscation to defeat, wherein the attacker uses a strategy of jumping into any existing code to execute any desired series of instructions.

Evasion Resistance

The evasion problem described by the AVLeak, Blackthorne et al research[2, 41] does not apply to this form of emulation, because the program is not truly being executed. Branches are forced randomly, taking away control-of-flow, which programs being emulated for natural behavior analysis rely on to evade. This approach by Blazytko et al gives the program no choice, it explores the possibility space broadly regardless of what decisions the program might naturally make. This "forced branching" method, which might aptly be referred to as "adversarial execution", would have a drawback of potentially hitting trap code (which might signal back to the command-and-control server not to send a malicious payload) if it were truly executing network activity. Rather, this method performs an unusual symbolic execution and has no need to make any network calls or execute the code in any real-sense, only explore branches and observe the instruction trace. The trap described will be described in further detail in a later section of this paper, but it is likely not an effective counter-measure against Program Synthesis.

Scanning Strategy

While the scanning capability presented by this paper focuses on an ability to scan kernel-mode processes, this does not preclude user-mode processes from the same scanning process. Both kernel and user modes are scanned; the kernel privilege is used to ensure access and integrity of the capability. Some components of the scan might even be executed in user-mode in the interest of stability or multi-threading. That said, code executed in user-mode must either be injected secretly or observed by a securely injected kernel mode process for external tampering by an attacker and should not execute any system calls (which might be compromised at the kernel-level).

When the scan is executed with the toolset described, Program synthesis is only one component of the scan. Once a code segment has been synthesized, there is no limit to the creative analysis which can be done by the scan to target known threat actors, look for IoCs in the system, or search for fundamentally malicious behavior regardless of matching any known signature.

Scan results need not and should not be treated as binary decisions. Ranges of suspicion should be communicated externally to a SIEM system. Thanks to ensured kernel-mode execution integrity, the scanner can also use a private network driver of guaranteed integrity to report scan data over the network (as part of an EDR system) without risk of being disrupted by a maliciously modified driver.

Performance

The research presented does suffer from high computational expense and therefore significant latency to synthesize full programs. In the presented form, it uses random branch exploration to analyze the program in extreme depth. This can be useful for passive scanning and threat hunting, but a faster version is needed for active scanning in order to recognize malicious code before it has an opportunity to execute.

This problem is very likely solvable through various optimizations. Blazytko et al synthesized random execution branches with random coverage of the entire program's tree structure. Instead, one can and should focus only on the path which can be executed in the immediate future based on the current state of the process at any given time. Programs need not be synthesized fully up-front. Just-In-Time (JIT) synthesis can be leveraged to partially analyze the program's near-future potential. The optimized process proposed is as follows:

1. Wait a TRNG-generated (securely unpredictable) random interval.
2. Read the instruction pointer.
3. Emulate a limited distance into the program's potential future.
4. Ensure no malicious signatures are ahead.
5. Allow the program to continue for another TRNG-generated random duration.
6. Ensure the Instruction Pointer is within an expected, emulated outcome range.
7. Send any executional deviation reports directly to the SIEM.
8. Repeat.

This analysis cycle does not attempt to emulate or scan every moment of a program's execution. Rather, it relies on an unpredictable interval of analysis. The attacker might be able to execute some malicious instructions, but it might just as easily be caught. The brevity of the true random delays in scanning (or in other words, the frequency of the scan) can be dynamically decided based on how much of the corresponding physical core's maximum performance is utilized.

The determining factors in the performance impact level such an approach will have on the system are the efficiency with which such analysis can occur and how much of the code executed loops upon itself. The overwhelming majority of code executing need not be re-analyzed. This is because most applications where users would be sensitive to latency follow a looping execution structure.

As an additional optimization, thanks to the black-boxed nature of this approach, one can parallelize a substantial portion of the analysis. When applicable (on devices with an integrated or external GPU) each segment of code may be treated as an operation in a GPU matrix or CPU thread, allowing for "synthesis" of millions of segments of code in parallel.

Overcoming Noise: I/O Entanglement

Program Synthesis considered; noise is still a problem. If there are 1000 decoy instructions between each instruction that contributes the program's true behavior signature, Program Synthesis will result in simplified noise. Because the strategy involves tracing instructions for some arbitrary sequence length, an attacker could simply counter the analysis by increasing the noise-to-signal ratio of the instructions.

It is important to make the distinction between decoy code and dead code because the defender can easily disregard branches that will not (or are very unlikely to) run. Decoy code must either execute or have a significant chance to do so based on environmental factors or random number generation. To reduce noise from decoy code this research proposes to track which code "entangles" with an eventual I/O event. Any code that is part of the real program behavior almost certainly is part of a series of branches that eventually somehow contributes to an I/O branch. Any instructions not meeting such criteria should be disregarded. This strategy should force an attacker to generate a high I/O signal-to-noise ratio, which is fundamentally problematic for the attacker as such I/O, even in random form, can become a red flag for the defender.

Because the overall strategy proposed involves analyzing ahead in the code only to an arbitrary distance, at any given instant the system may not yet know which of the current instructions are in some way contributory toward an I/O event. However, this challenge can be overcome by tracking contribution and allow the program to proceed, since without an I/O event, the program cannot produce an effect. Once an I/O event is encountered, the system can look backwards and see which previously executed instructions contributed in some way, directly or indirectly, toward the I/O event's execution. A hypothesis is that such an approach will allow the analysis system to ignore noise instructions even where they outnumber genuine instructions drastically.

For the sake of clarity, “synthesis” will hereafter mean Blazytko et al’s version in addition to the described I/O entanglement noise reduction.

Script Scanning

There is no need for special deobfuscation of scripts. JavaScript, PowerShell, Bash, AutoIt, and each of the thousands of interpreted languages are nothing more than data fed to a process. Based on the data (the script), the PowerShell or Bash or Chrome process will internally branch a specific way and form a series of state transitions. This can be analyzed through Program Synthesis just like any standalone executable binary. There is no need for special handling, however the open source code released by the researchers is not a polished product and might need expansion for the forced branching analysis to take command input as an initial state to branch outward from, rather than indiscriminate program tree exploration.

Payload Isolation

Two options exist for isolating a signature for the attacker's source-code:

1. Automated: For less severe repeat attacks, the payload is automatically isolated with no need for human analysis. To avoid standard hashing signatures, the repeat attacker must-utilize multiple re-obfuscated strains of their malware. Once the same payload has been captured and synthesized two or more times, the malicious elements can be easily extracted from matching logic, where any non-matching synthesized logic is less interesting as a signature and in most cases just random noise added during the obfuscation process.
2. Manual: For more severe, urgent attacks, a malware analyst performs standard deobfuscation and reduction of the payload to its malicious logic, before synthesizing the result and using it as a signature.

Once the synthesized logic is achieved via either method, a database containing millions of synthesized benign executables is referenced to remove non-unique code segments from the synthesized malware, before storing what is left as a final malware signature.

Even against ROP-based obfuscation where mostly re-used code is involved, there will remain components of the payload which are used to build the ROP gadgets and those will serve as effective signatures.

Establishing State-of-the-Art Offense

An in-depth coverage of the tradecraft involved in effective payload obfuscation and behavior analysis evasion (which would require an entire research paper) is foregone in favor of a brief example of a cyber weapon this paper might propose to subvert. Many potential features are skipped for brevity. This section is intended to ensure that efforts are focused on a minimum agreed upon baseline minimum threat model. It likely offers no significant innovation to existing state-of-the-art attack frameworks and can be skipped.

User configuration

- User assigns its developed or acquired exploits to vulnerable services.
- User assigns its developed or acquired payloads to OS environments in executable binary format.
- User assigns target entry IPs.
- The C2 server does the rest.

Armament

- The user is a nation state-attacker.
- Zero-day exploits for Remote Code Execution (RCE) with Privilege Escalation (PE) to ring-0 for Windows, Linux, and an assortment of Firewall and Router operating systems, in addition to major smartphone vendors are supplied.

Natural Path to the Domain Controller

- Custom versions of Bloodhound and Mimikatz work together to enumerate the network targeting the domain controller. Obfuscation in the next sections will ensure these tools are not detectable through

I/O signatures (which is how both are typically caught - their signature I/O patterns create distinguishable IoCs).

- Only natural traffic flow is leveraged. Once an endpoint is compromised, further exploitation is only delivered through the compromise of an internal application in order to hide lateral movement within naturally occurring network exchanges.
- The attacker provides the payload containing the desired effect.
- The attacker's exploits are delivered dynamically only if the corresponding vulnerable services mapped to them are within reach, so the attacker can purchase or develop exploits to a wide range of services that might be within the targeted network(s), and only risk burning the ones that are used.
- Once the Domain Controller is compromised, Windows update services are compromised in order to create effects on the remainder of the domain.

Artifact Trimming

- The C2 server emulates the payload and flags I/O which cannot effectively be obfuscated for modification. It warns the attacker to fix certain artifact which cannot be automatically obfuscated.
- If a payload prints "I'm evil" to `stdout`, that cannot be automatically obfuscated away without potentially breaking the program. The attacker must either heed these flags, make modifications, or ignore them and continue.
- Any system calls that have no obfuscation potential are suggested for re-consideration.
- Most system calls in the original payload source are achieved through a kernel-mode component recreating the effects of system calls through alternative means in order to create the desired effects without logging or tracing of behavior.
- Any memory activity is either a decoy and out in the open or encrypted / encoded with a key only present in-register, making memory scanning of the malicious payload difficult.
- As much of the payload logic as possible is kept in the registers, only streaming memory values when necessary.
- To avoid this behavior creating a potential fingerprint, any encrypted data is encoded into a benign data format while in memory.
- To evade potentially undocumented CPU microcode or JTAG register scanning approaches which might be leveraged by nation-state targets, register values are obfuscated.

Polymorphism

- To avoid a fingerprint developed for one instance of the payload being leveraged against any others, each time the payload spreads, it re-obfuscates both itself and its obfuscation code in-place.

Automated Network Intermediary Routing

- Any web request targets are changed to intermediate IPs. Rather than querying "supervillain.com", the attack suite will direct this query through some benign intermediary. No outbound traffic will target a C2 server or any suspicious service directly. Web traffic logs are reviewed on a host-by-host basis within the payload to determine which traffic targets are usual. Social networks and public forums are contacted over HTTPS with data encrypted and encoded into benign format.
- C2 server observes the potential end-points, decodes and decrypts the data, then forwards the traffic to its intended target, then sends the response in the same manner.
- Targets which do not normally send outbound traffic stream exfiltration through another compromised endpoint which can exfiltrated the data in a natural manner.

Emulator Evasion

- Then the C2 server wraps the payload in emulation evasion logic by doing black-box analysis against the most recent release of every popular behavior emulation tool[2].
- Inconsistencies in implementations of the x86-64 spec[15] by processors and every mainstream emulator are leveraged to diverge actual execution from the defender’s emulated execution trace.

Binary Obfuscation

- The C2 server then performs advanced binary obfuscation on the resulting executable for every Nth attack entry-point in sweeping attacks (every payload may be uniquely obfuscated, or every 100th payload sent may be, depending on how many versions of the attack are pre-computed for delivery), so the payload delivered to each target is potentially unique:
- C2 uses a genetic-algorithm-based obfuscation program. Reinforcement learning is leveraged.
- C2 traverses executable branches, treating segments of code as black boxes, where it must arrive at the same outcome through different means limited not only to system calls but even register activity.
- Return Oriented Programming (ROP) is introduced as a form of obfuscation[3].
- Any system calls are creatively achieved in alternative ways to obfuscate the system call I/O fingerprint. Multiple alternatives are introduced as pseudo random branching conditions.
- If the attacker marks the stage as running with kernel mode privilege, C2 achieves the system call's objective without using system calls at all.
- Random system calls are introduced between any calls made by the original payload (introducing noise into the I/O analysis channel relied on heavily by EDR solutions)
- Decoy logic noise is of course also introduced. Not dead code, however; each decoy instruction may conditionally execute.
- All statistical usage (patterns: CPU, memory, storage, power) is randomized, evading machine-learning based resource usage pattern detection demonstrated in research.
- Each of the above binary obfuscation techniques are designed to either utilize some red-team academic research I have read, or counter blue-team detection strategies.

Defeating Forced Branching: Many-staged Delivery

- Next the C2 server defeats forced branching emulation, by delivering traps in many stages:
- If the attacker is detecting the emulation / VM and behaving innocently, the defender must either make the VM more believable (an unreliable game of cat and mouse, defeated easily)[2, 41] or force alternate branching.
- To counter this, before the payload branches toward downloading the malicious code, the program must progress past trapped branches that have a 50% pseudo random chance of executing. This forces the defender not to discount them as decoy code, force the branch to execute, and after enough of these branches are forced, each calling more code from the command-control server, the C2 server

can see that the odds of so many of these 50% chance branches all executing naturally is extremely low. It then withholds the malicious payload by sending a final benign code-set which will not call for any further code. Later when the program is run on the true target without branch forcing, the payload will be sent instead.

Automated pre-flight test

- The resulting stages are then executed in a VM against every popular EDR and AV solution and (optionally) sent to VirusTotal. If the result is failure, the stages are re-processed, and the user again prompted to fix any previously flagged I/O artifacts which may be triggering detection.
- IDS, IPS, and endpoint heuristic rules (Snort, YARA, Sigma, etc) are automatically generated against the payload and its network behavior, and further obfuscation of the exposed signatures is done before re-testing until as many signatures as possible have been eliminated from the payload.

And More

Countless potential approaches exist, and each must be considered by the defense researcher. Hiding the payload on a location in the filesystem which the filesystem driver is patched to mis-read might be a tactic. That is just one example of many more.

Private Infrastructure National Attack Vector

This section will outline the reasons leaders should be concerned about a critical attack vector against private infrastructure and a potential use-case for the toolkit described in this paper to further the national-security objective of protecting it. The plug-and-play nature of the proposal makes it a suitable candidate for addressing a private industry disruption attack vector.

Sitting POTUS Biden has dictated that further action be taken on this front[38]:

"The United States faces persistent and increasingly sophisticated malicious cyber campaigns that threaten the public sector, the private sector, and ultimately the American people's security and privacy. The Federal Government must improve its efforts to identify, deter, protect against, detect, and respond to these actions and actors. The Federal Government must also carefully examine what occurred during any major cyber incident and apply lessons learned. But cybersecurity requires more than government action. Protecting our Nation from malicious cyber actors requires the Federal Government to partner with the private sector." - Executive Order 14028, Improving the Nation's Cybersecurity, 2021

The private sector is widely accepted as a severe threat vector and has been publicly reported as such[22]. This concern has been officially validated through a 2021 Threat Assessment by the Intelligence Community:

"A Russian software supply chain operation in 2020, described in the cyber section of this report, demonstrates Moscow's capability and intent to target and potentially disrupt public and private organizations in the United States." - Annual Threat Assessment of the US Intelligence Community, 2021

Unlike air, space, sea, or land, the cyberspace battlefield provides military leaders with little room for maneuver when it comes to defending the nation's borders. The cyber battle-ground is uniquely exposed. Logic bombs are transported over encrypted communication channels to detonate at the doorstep of private enterprise. The privacy rights afforded citizens by the U.S. Constitution ensure this. On the battlefield of cyber warfare, borders are redefined. Instead of physical boundaries, these borders exist within the networks of not only government agencies, but also private industry. Cyber borders have neither the advantage of terrain, nor armies, nor weapon systems, except those imposed by civilians. Those civilian measures are adequate to mitigate civilian threats, but do not address nation-state threats. Nor should civilians be expected to fund such military-grade defenses.

There appears to be no consensus on the extent of damage a nation-wide cyber attack on American soil might produce, nor the number of deaths projected. This would depend on many factors including the vulnerability of medical equipment in hospitals, reliability of generators, how long it might take for the electric grid to recover, the effects of oil shortages, whether retailers can transition to a cash transaction model, whether their suppliers will be able to recover in a timely manner, how soon banks will be able to create access to cash for the people needing it, and how people will react to the shortages that follow. This paper will not thoroughly explore this many faceted topic, but rather outlines key considerations.

Entire industries need not be attacked for a significant national disruption to occur. For one example of how a single failure in any supply chain can create a long cascade of disruptive effects: A single payroll vendor, if disrupted, might disrupt the operation of tens of thousands of client organizations which rely on them to track employee payment. If operations are disrupted in a significant number of suppliers as a result, resulting shortages might cascade further down the line, affecting companies dependent on those supplies, and so on.

The widely used Active Directory protocol used to administrate large networks makes attacking private industries a potentially very automated process, meaning a single cyber weapon might be able to dynamically enumerate a wide range of critical networks. For a sufficiently resourceful attacker armed with enough zero-day exploits, modern defense systems leveraged by many private and event government organizations can be rapidly attacked. These attacks are typically done slowly and quietly, however this is because attacks are typically designed by profit-seeking criminals to be as cheap as possible, meaning the attackers do not have the advantage of wide-access to expensive zero-day exploits. Well-funded attackers with non-financial incentive have no such limitation.

The most obvious apparent solution to this problem is one of retaliation. If a near-peer adversary perpetrates this attack, the U.S. might retaliate severely. For this reason, leaders might believe the best defense is a good offense. This solution is insufficient under two scenarios:

- Active war with an adversary.
- Plausible deniability.

While active war should be an obvious example, plausible deniability is more nuanced and warrants elaboration. If the U.S. or an allied near-peer designs a modular cyber weapon which streamlines the process of executing an automated attack against any network model, that toolset might be leaked through insider threat or stolen by a foreign intelligence agency or military power. This power might never use the tool or directly attack U.S. interests with it and claim plausible deniability, blaming the U.S. or its allies for losing control of their cyber weapons. But with this tool available, a terrorist organization might only need to purchase, for example, 10 new zero-day exploits to carry out the attack. It is not impossible to imagine a future terrorist organization with sufficient funding to leverage such a leaked toolset, purchase (or covertly receive) the exploits necessary to arm the tool with and carry out such an attack against private infrastructure in the U.S. Terrorist organizations have historically been unthwarted by the threat of kinetic retaliation.

Currently, the defensive cyber-security posture of the U.S. Government as it relates to private industry appears to be an academic and indirect one: Invest in organizations which research the problem and present standards, best-practices, and guidance for companies to leverage. Private enterprise has been treated as the target of advice, standards, information, and regulation, but rarely as an ally to equip with a powerful toolset. As an exception, the NSA did this successfully with the Ghidra project. However, it may be in the interest of national security to provide more substantial tools to achieve a certain threshold of defended private infrastructure in order to mitigate loss-of-life in the case of a nation-wide cyber attack.

The problem faced is that private companies naturally see the issue of cyber-security just as they do any other: One of cost and benefit. This is not a barrier to national security, only one which must be understood by a leader who desires to strengthen it. The first paragraph of the Executive Summary for the National Institute of Standards and Technology (NIST) Framework lays out succinctly this cost-benefit relationship:

> "The United States depends on the reliable functioning of critical infrastructure. Cyber security threats exploit the increased complexity and connectivity of critical infrastructure systems, placing the Nation's security, economy, and public safety and health at risk. Similar to financial and reputational risks, cyber security risk affects a company's bottom line. It can drive up costs and affect revenue. It can harm an organization's ability to innovate and to gain and maintain customers." - NIST Framework

Research papers focus on achieving the most optimal cost, benefit dynamics on the matter[39]. Unfortunately, what is optimal for private industry can be far from optimal for national defense objectives, as is highlighted in the quoted paragraph. When a company is disrupted, it is an investor's problem. When an industry is disrupted, it is the nation's problem.

The most obvious problem is one of technology. To date, there does not exist a publicly available tool capable of effectively mitigating a nation-state level cyber attack. State-of-the-art cyber attacks involving access to modern operating systems at ring 0 usually the domain of nation-state attackers. Accordingly, most private research and development efforts are aimed at less sophisticated, more common, and easier-to-solve attack vectors. Therefore, a lack of solution in this rapidly evolving problem-space is unsurprising.

For the nation to be secured against this threat, some minimum threshold of private infrastructure in addition to all public utility infrastructure must be secured with the capability to recover from a state-of-the-art national cyber attack within some time limit. This might also serve as a free, public option for meeting mandated compliance, especially with the U.S. Government passing new mandates[19] for defense against these threats by critical private industries. This research does not attempt to quantitatively describe the threshold of operational grocery stores, gas stations, oil pipelines, and other supply chain members a given state needs for its people to survive; the Department of Defense can surely obtain such metrics.

With the capabilities suggested by this proposal developed by the Department of Defense and leveraged through voluntary cooperation from private infrastructure owners, perhaps incentivized by some form of subsidy, the U.S. Government can harden private industries against this threat vector. Following the attack described, or perhaps even immediately after the cyber weapon leakage, the NSA would use Program Synthesis to burn the tool and publish the signature, equipping private enterprise and government administrators around the nation with the ability to detect and prevent the attack reliably.

Requirements

To bring the project beyond concept phase, a dedicated tool development team should work through an initial exploratory stage, with at least one senior developer with experience developing drivers overseeing the project. Additionally, access to a state-of-the-art payload obfuscation tool to test the obfuscation-defeat capability against is a requirement, otherwise the team will have to first develop one based on the specifications provided. Regular access to seasoned computer engineering, software engineering, and offensive exploitation SMEs should be available.

Before a hardware development effort is invest in, a virtual proof of concept should be completed and tested through emulation to ensure performance requirements are met. An initial proof of concept project aimed at supporting a small subset of potential environments is recommended. A complete toolset will require multiple software teams working in parallel to develop custom variants of the kernel injection code for each kernel / OS the project will support. The custom RAM component will likely be the most expensive research and development effort because the electrical engineering behind DDR memory is among the most sophisticated in computer engineering. A highly experienced computer engineering team will be required.

Sources

1. *Breaking State-of-the-Art Binary Code Obfuscation via Program Synthesis*, Blazytco et al (https://i.blackhat.com/briefings/asia/2018/asia-18-Blazytko-Breaking-State-Of-The-Art-Binary-Code-Obfuscation-Via-Program-Synthesis-wp.pdf)
2. *AVLeak: Fingerprinting Antivirus Emulators Through Black-Box Testing*, Blackthorne et al (https://www.usenix.org/system/files/conference/woot16/woot16-paper-blackthorne.pdf)
3. *Intertwining ROP Gadgets and Opaque Predicates for Robust Obfuscation,* Nakanishi et al *(*https://arxiv.org/pdf/2012.09163.pdf*)*
4. *ProvenCore: Towards a Verified Isolation Micro-Kernel, Lescuyer (*https://www.provenrun.com/wp-content/uploads/2015/01/Prove-Run-ProvenCore-Towards-a-Verified-Isolation-Micro-Kernel.pdf*)*
5. *An Empirical Assessment of Endpoint Detection and Response Systems against Advanced Persistent Threats Attack Vectors*, Karantzas et al (https://www.mdpi.com/2624-800X/1/3/21)
6. *Evading Machine Learning Malware Detection*, Anderson et al (https://www.blackhat.com/docs/us-17/thursday/us-17-Anderson-Bot-Vs-Bot-Evading-Machine-Learning-Malware-Detection-wp.pdf)
7. *HLMD: a signature-based approach to hardware-level behavioral malware detection and classification, Bahador et al (*https://link.springer.com/article/10.1007/s11227-019-02810-z)
8. *Integrating cost–benefit analysis into the NIST Cyber security Framework via the Gordon–Loeb Model*, Gordon et al (*https://academic.oup.com/cyber security/article/6/1/tyaa005/5813544)*
9. *Project: Rosenbridge, Hardware Backdoors in x86 CPUs,* Domas et al *(https://i.blackhat.com/us-18/Thu-August-9/us-18-Domas-God-Mode-Unlocked-Hardware-Backdoors-In-x86-CPUs-wp.pdf)*
10. *The Geometry of Innocent Flesh on the Bone: Return-into-libc without Function Calls (on the x86)*, Shacham (https://hovav.net/ucsd/dist/geometry.pdf)
11. *Rootkits and Bootkits: Reversing Modern Malware and Next Generation Threats,* Matrosov, ISBN-13: 978-1593277161 (https://www.amazon.com/Rootkits-Bootkits-Reversing-Malware-Generation/dp/1593277164)
12. Intel 64 and IA-32 architectures software developer's manual combined volumes: 1, 2A, 2B, 2C, 2D, 3A, 3B, 3C, 3D, and 4 (https://software.intel.com/content/www/us/en/develop/download/intel-64-and-ia-32-architectures-sdm-combined-volumes-1-2a-2b-2c-2d-3a-3b-3c-3d-and-4.html)
13. NIST Framework for Improving Critical Infrastructure Cyber security (https://www.nist.gov/system/files/documents/cyberframework/cyber security-framework-021214.pdf)

14. *The Memory Sinkhole - Unleashing An X86 Design Flaw Allowing Universal Privilege Escalation*, Domas @ Blackhat 2015 (https://www.youtube.com/watch?v=lR0nh-TdpVg)
15. *Breaking the x86 Instruction Set*, Domas @ Blackhat 2017 (https://www.youtube.com/watch?v=KrksBdWcZgQ)
16. *Hardware Backdoors in x86 CPUs*, Domas @ Blackhat 2018 (https://www.youtube.com/watch?v=_eSAF_qT_FY)
17. *Breaking State-of-the-Art Binary Code Obfuscation via Program Synthesis*, Blazytco @ Blackhat 2018 (https://www.youtube.com/watch?v=0SvX6F80qg8)
18. *AVLeak: Fingerprinting Antivirus Emulators for Advanced Malware Evasion,* Blackthorne et al @ Blackhat 2016 (https://www.youtube.com/watch?v=a6yOwvFds78)
19. *DHS Announces New Cyber security Requirements for Critical Pipeline Owners and Operators, Department of Homeland Security (*https://nsarchive.gwu.edu/document/23932-department-homeland-security-press-release-dhs-announces-new-cyber security*)*
20. *NIST.gov (https://www.nist.gov/)*
21. *What is Endpoint Detection and Response?* (EDR), McAfee (https://www.mcafee.com/enterprise/en-us/security-awareness/endpoint/what-is-endpoint-detection-and-response.html)
22. *The next 9/11 will be a cyberattack, security expert warns*, CNBC (*https://www.cnbc.com/2018/06/01/the-next-911-will-be-a-cyberattack-security-expert-warns.html*)
23. *Ghidra,* National Security Agency (https://www.nsa.gov/resources/everyone/ghidra/)
24. *John von Neumann*, Google Scholar (https://scholar.google.com/citations?user=6kEXBa0AAAAJ&hl=en-GB)
25. *Theory of self-reproducing automata*, John von Neumann (https://archive.org/stream/theoryofselfrepr00vonn_0/theoryofselfrepr00vonn_0_djvu.txt)
26. *Computer Viruses - Theory and Experiments (1984)*, Fred Cohen (http://web.eecs.umich.edu/~aprakash/eecs588/handouts/cohen-viruses.html)
27. *Computer Viruses (1986)*, Fred Cohen (http://www.all.net/books/Dissertation.pdf)
28. *Inventors and Inventions, Volume 4*, Marshall Cavendish (https://books.google.com/books?id=YcPvV893aXgC)
29. *1987*, VirusList.com (https://web.archive.org/web/20071107040723/http://www.viruslist.com/en/viruses/encyclopedia?chapter=153311150)
30. *Formal Verification of Hardware Components in Critical Systems*, Kamran et al (https://www.hindawi.com/journals/wcmc/2020/7346763/)

31. *CVE-2017-5689* (https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-5689)
32. *One-pixel Attack Fools Neural Network*, Hackaday (https://hackaday.com/2018/04/15/one-pixel-attack-fools-neural-networks/)
33. *One pixel attack for fooling deep neural networks*, Jiawei Su et al (https://arxiv.org/abs/1710.08864)
34. *Annual Threat Assessment of the US Intelligence Community, 2021,* Office of the Director of National intelligence (https://www.dni.gov/files/ODNI/documents/assessments/ATA-2021-Unclassified-Report.pdf)
35. *Malware Buried Deep Down the SPI Flash: Sednit's First UEFI Rootkit Found in the Wild* (https://i.blackhat.com/eu-18/Wed-Dec-5/eu-18-Boutin-Vachon-Malware-Buried-Deep-Down-The-SPI-Flash-Sednits-First-UEFI-Rootkit-Found-In-The-Wild-wp.pdf)
36. *System Management Mode deep dive: How SMM isolation hardens the platform* (https://www.microsoft.com/security/blog/2020/11/12/system-management-mode-deep-dive-how-smm-isolation-hardens-the-platform/)
37. *Advanced x86: BIOS and System Management Mode Internals SPI Flash Protection Mechanisms*, Legbacore (https://opensecuritytraining.info/IntroBIOS_files/Day2_03_Advanced%20x86%20-%20BIOS%20and%20SMM%20Internals%20-%20SPI%20Flash%20Protection%20Mechanisms.pdf)
38. *Executive Order 14028 of May 12, 2021 Improving the Nation's Cybersecurity,* federalregister.gov (https://www.federalregister.gov/documents/2021/05/17/2021-10460/improving-the-nations-cybersecurity)
39. I*ntegrating cost–benefit analysis into the NIST Cybersecurity Framework via the Gordon–Loeb Model*, Gordon et al (https://academic.oup.com/cybersecurity/article/6/1/tyaa005/5813544)
40. *Detecting Hyperjacking in cloud based virtual environment*, Dennyson et al (http://sersc.org/journals/index.php/IJAST/article/download/15440/7789/)
41. *AVLeak: Fingerprinting Antivirus Emulators for Advanced Malware Evasion*, Blackthorne @ Blackhat 2016 (https://www.youtube.com/watch?v=a6yOwvFds78)
42. *Zero-Trust Networks: The Future Is Here - SANS Blue Team Summit 2019*, SANS Institute (https://www.youtube.com/watch?v=EF_0dr8WkX8)
43. *SANS Webcast - Zero Trust Architecture*, SANS Institute (https://youtu.be/5sFOdpMLXQg)

44. *Formalising Trust as a Computational Concept*, Marsh (https://dspace.stir.ac.uk/bitstream/1893/2010/1/Formalising%20trust%20as%20a%20computational%20concept.pdf)

45. *CALCULATING ATRUST SCORE, Patent No US 9,578,043 B2*, Mawji et al (https://patentimages.storage.googleapis.com/1f/9a/ac/45022101d8135e/US9578043.pdf)