On the Feasibility of Detecting Software Supply Chain Attacks

Xinyuan Wang, *Member, IEEE*

***Abstract*—The Supply chain attack is the stealthy and sophisticated cyberattack that aims to compromise a target by exploiting weaknesses and vulnerabilities in its supply chain. Recent supply chain attacks (e.g., SolarWinds attack) have compromised some of the most secured IT infrastructures of government agencies and enterprises. The European Union Agency for Cybersecurity, ENISA, has predicted that there will be 3 times more supply chain attacks in 2021 than in 2020.**

In this paper, we look into the problem of supply chain attacks, the challenges of defending software supply chain attacks. We analyze what it takes to effectively prevent software supply chain attacks, and show that it is indeed feasible and practical for the customers to detect certain software supply chain attacks. We propose an information flow based detection approach that enables end users to detect many software supply chain attacks without dealing with any of the underlying software suppliers.

***Index Terms*—Supply chain attack, advanced persistent threat, cyber defense.**

I. Introduction

In his 1984 ACM Turing Award lecture [1], Ken Thompson, one of the creators of the Unix operating system, had illustrated the first software supply chain attack via a trojaned C compiler. In the past few years, we started to see real world supply chain attacks that compromised global enterprises and caused massive data breaches. In 2013, cyber criminals compromised the data network and POS systems of Target via the stolen network credential of Target's HVAC service provider Fazio Mechanical Services [2], stole 40 million credit/debit cards of Target customers, and costed Target \$202 millions. According to the 2019 Cost of a Data Breach Report[3], the average cost of massive data breach of 50 million records is \$388 million.

The 2020 SolarWinds hack is a sophisticated supply chain attack that has breached tens of thousands of SolarWinds Orion customers worldwide. Specifically, the adversary had gained access to SolarWinds' software build system by September 2019, and had planted malware into Orion software updates in March 2020. 18,000 Orion customers including various US government agencies (e.g., DoD, DHS, DOJ) and many Fortune 500 businesses have downloaded and installed the compromised Orion updates that contain stealthy backdoors. Since SolarWinds Orion is the network monitoring and management system that oversees the whole network, hosts and servers in the operation environment, the backdoors in the Orion system gave the adversary stealthy access to all the mission critical networks and systems being monitored. It has been reported that adversary was able to access the email system used by the highest-ranking officials in the Treasure Department. "Having accessed data of interest, they encrypted and exfiltrated it" [4]. Now `solarleaks.net` is selling gigabytes of source code of Microsoft, Cisco, SolarWinds and FireEye that were stolen and exfiltrated during the 2020 SolarWinds breach for \$1,000,000 [5]. Microsoft president Brad Smith considered SolarWinds attack as "the largest and most sophisticated attack the world has ever seen" [6].

Besides SolarWinds Orion, other widely used software such as Microsoft exchange server [7], Kaseya VSA remote monitoring and management SaaS [8] have recently been compromised to launch supply chain attacks and ransomware attacks.

While existing cyber defense mechanisms such as firewall, antivirus, intrusion detection and prevention system (IDS/IPS), authentication and code signing are very useful, they have been shown to be not effective against software supply chain attacks. According to the CISA spokeswoman Sara Sendek, none of existing deployed intrusion detection or prevention systems – including the U.S. government's multi-billion dollar detection system, Einstein, was able to detect the 2020 SolarWinds breach [9].

After analyzing 24 software supply chain attacks that happened between January 2020 and early July 2021, the European Union Agency for Cybersecurity (ENISA) has recently warned that current cyber defenses against software supply chain attacks are not effective and has predicted that the number of massive supply chain attacks in 2021 is likely to be 4 times of that in 2020 [10]. Therefore, there is a pressing need to develop novel and practical defensive measures to detect, mitigate and respond to potential supply chain attacks in the near future.

In this paper, we look into the problem of software supply chain attacks, the technical challenges in defending against software supply chain attacks, and analyze what it takes to defend against software supply chain attacks. Based on tracking information flows from strategic locations (e.g., mission critical server), we propose a practical and signature-less supply chain attack detection approach that enables enterprises and organizations to detect novel supply chain attacks without dealing with any of the underlying software and hardware suppliers.

II. Supply Chain Attacks

The *supply chain* is the ecosystem of organizations, people, processes, activities and resources involved in creating, distributing and supplying products or services to customers. The *supply chain attack* is the cyberattack that attempts to compromise a target by exploiting weaknesses and vulnerabilities in its supply chain. According to the 2021 European Union Agency for Cybersecurity ENISA Threat Landscape for Supply Chain Attacks [11], a supply chain attack starts

Xinyuan Wang is with the Department of Computer Science, George Mason University, Fairfax, VA 22030, USA. email: xwangc@gmu.edu

with the attack on one or more suppliers in the supply chain. Once the attack on the suppliers has compromised supplier assets (e.g., some software supplied to some customer by some supplier), it will targets the customers of the suppliers (e.g., users of the software) aiming to compromise the customer assets (e.g., sensitive data of the customer).

In a supply chain attack, the supplier assets targeted by the attack could include software, hardware, data, processes and people. Most of the 24 supply chain attacks (from January 2020 to early July 2021) analyzed by the 2021 ENISA Threat Landscape for Supply Chain Attacks report [11] are software related supply chain attacks. Specifically, 62% of those 24 supply chain attacks have used malware; 50% have been attributed to well-known APT groups (e.g., APT 29) and 58% have aimed at stealing customers' data. In this work, we focus on analyzing and detecting software supply chain attacks (e.g., SolarWinds attack [12], [13]) because of their prevalence and far reaching impact.

III. Challenges in Defending against Supply Chain Attacks

A. Supply Chain Attack Prevention

Ideally, we want to prevent supply chain attacks from happening in the first place. Traditional cyberattack prevention mechanisms are built upon the trust of certain credentials. For example, firewalls generally trust outbound traffic more than inbound traffic, and it only allows inbound traffic to pass after it has been properly authenticated. Authentication – including two factor authentication (2FA), and access control mechanisms use various credentials such as 1) something you know (e.g., secret password, private key); 2) something you have (e.g., physical token, cell phone); 3) who you are (e.g., retina, fingerprint biometrics) to authenticate users, systems and digital artifacts. In addition, critical code and code updates from many software suppliers (e.g., Microsoft, SolarWinds) are digitally signed to make sure only authentic code and code updates from the trusted vendors will be installed and executed in customers' systems. In the current cybersecurity paradigm, an user, system or digital artifact is considered trustworthy once it has been properly authenticated and has passed existing cybersecurity checks.

While traditional preventive measures (e.g., firewall, authentication) are quite effective in preventing cyberattacks from untrusted sources, they are not effective against insidious supply chain attacks that exploit the inherent trust customers generally have in the suppliers they choose.

The 2020 SolarWinds supply chain attack demonstrated a particular challenging case in preventing the supply chain attacks. Once the adversary had compromised the SolarWinds Orion production system and had planted malicious code in Orion software updates, those malicious Orion updates would eventually be digitally signed by SolarWinds before distribution. From customers' perspective, those digitally signed malicious Orion updates are authentic thus trustworthy given their trust in SolarWinds Orion products. Once installed, the digitally signed and trusted but malicious Orion updates would make the trusted and privileged Orion network monitoring system malicious!

From defender's perspective, supply chain attacks have essentially increased the attack surface by orders of magnitude. In order to protect a mission critical system from supply chain attacks, it is no longer enough to just secure the mission critical system itself, but we have to secure the development, production, authentication, distribution and updates of all the underlying components in the mission critical system. Note, contemporary software systems are large and sophisticated and they often have complicated dependency on many third party libraries (e.g., OpenSSL). Furthermore, certain widely used open source software packages may not be actively maintained by volunteers and certain widely used software packages are closed source. Therefore, it is infeasible for any owner of any mission critical system to enforce or compel each supplier of each software component to have the same cybersecurity scrutiny needed. This inevitably leaves certain supplier more vulnerable than the rest.

Once the adversary has compromised some trusted but vulnerable supplier and has generated digitally signed malicious updates, no existing cyber defense can prevent such insidious supply chain attacks as there is no way for the customer's cyber defense system to tell if a digitally signed update from a trusted vendor is malicious or not.

Therefore, supply chain attacks have forced us to abandon the level of trust we use to have in those reputable and supposedly trustworthy suppliers. The open question is how much trust, if any, we should have in those supposedly trustworthy suppliers.

B. Supply Chain Attack Detection

If we can not prevent supply chain attacks from happening, we want to detect those supply chain attacks at customer's mission critical systems, hopefully in real-time.

Supply chain attacks often involve application specific exploits on one or more software components of the target system. Detecting application specific exploits, however, are challenging due to the need of the intimate knowledge of the application. Given the enormous number and size of applications (e.g., open source, closed source), it is simply impossible for any cyber defense mechanism to have the intimate knowledge of all applications. Furthermore, detecting previously unknown application specific exploits is exceedingly difficult due to its inherent dependency on the yet-to-know vulnerabilities of the application. While an application may function exactly as expected under normal conditions with normal inputs, it could behave, with abnormal inputs, in a way no one has ever imagined. For example, Bash shell, which was first released in June 1989, has been used as the default login shell for most Linux distributions and all MacOS releases before Catalina. After 25 years of extensive use, Bash had been found to have a family of security vulnerabilities – Shellshock [14] that allow attackers to execute arbitrary commands and gain unauthorized access to many Internet-facing services (e.g., web servers) that use Bash to process the incoming requests. Before the Shellshock vulnerability was discovered in September 2014, no deployed cyber defense mechanism had ever detected any exploit of the Shellshock vulnerability.

Most existing intrusion detection (IDS) systems – including U.S. government's multi-billion dollar detection system, Einstein, use signatures or behavior patterns of known attacks to detect cyberattacks. Detecting known exploits of known application specific vulnerabilities are also technically challenging as the adversaries could use polymorphism to disguise their application specific exploits. According to the Symantec Internet Security Threat Report Volume 24 [15], there were 669 millions new malware variants found in 2017 alone. That means on average over 1.83 million new variants appeared every day. Neither signature based nor machine learning based cyber defense approaches could keep up with over 1.83 million new malware variants per day! In fact, all of the currently deployed intrusion detection systems – including the multi-billion dollar Einstein detection system, failed to detect the 2020 SolarWinds supply chain attack [9].

Since contemporary mission critical systems are enormously large (e.g., consists of millions of lines of code) and usually have very sophisticated dependency on many third party software packages provided by different suppliers, it is simply impossible for any cyber defense to known the inner workings of all the software components provided by all suppliers. To address the technical challenges in detecting polymorphic and application specific exploits, we need to develop novel detection capability that 1) is independent from various suppliers; 2) requires no signature or prior knowledge of the attack.

IV. An Information Flow based Detection of Supply Chain Attacks

In this section, we describe a novel supply chain attack detection approach, based on the inherent information flows in supply chain attacks, that is independent from all the suppliers and requires no signature or prior knowledge of the supply chain attack.

A. Information Flows in Supply Chain Attacks

No matter what application specific exploit a supply chain attack has used to compromise the final target, there is always some information flow involved in the supply chain attack. Besides the infiltration information flow that enabled the attacker to compromise the target, almost all supply chain attacks have some sort of data exfiltration that carries the sensitive information the attacker wants to steal. As shown in Figure 1, the attacker in a supply chain attack can compromise some data source and surreptitiously exfiltrate data from the data source while legitimate users may access the same data source at the same time. For example, the 2013 Target supply chain attack has exfiltrated the credit/debit card information of 40 million Target's customers from Target's POS devices. Furthermore, sophisticated supply chain attacks could use command and control (C&C) channel to allow remote attackers to explore and manipulate the compromised mission critical targets. For example, the 2020 SolarWinds supply chain attack connected to the command and control (C&C) centers 12-14 days after being installed. Apparently the data exfiltration and C&C traffic only exist when the supply chain attack is going on. Such unique information flows form

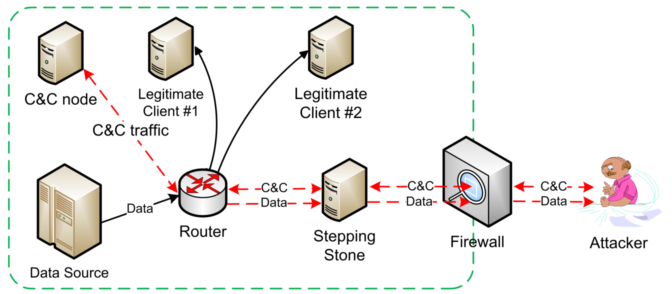

Fig. 1. The Attack Information Flows During Supply Chain Attacks

the very foundation of our information flow based supply chain attack detection.

Tracking the data exfiltration and C&C traffic is not trivial. The 2013 Target data breach and the 2020 SolarWinds data breach all have encrypted the stolen data before data exfiltration [4]. In addition, SolarWinds supply chain attack laundered its C&C control traffic through a number of Amazon cloud virtual machines as stepping stones before reaching its C&C center. Therefore, one key to detect sophisticated supply chain attacks is the capability to reliably identify and track those suspicious, encrypted and laundered data exfiltration and C&C traffic from the target.

B. Identifying and Tracking Attack Information Flows

As shown in Figure 1, there could be legitimate data access while the supply chain attack is exfiltrating data and laundering encrypted C&C traffic across a number stepping stones. To effectively identify and track the encrypted and laundered information flows of the supply chain attacks, we need capabilities to

- distinguish any illegal data access from all legitimate data accesses of specified data sources
- track encrypted traffic across stepping stones

We have the following observations from the reported supply chain attacks

1) abnormal data exfiltration from mission critical data sources to new destinations happened only when supply chain attacks (e.g., 2013 Target breach, 2020 SolarWinds breach) were going on.
2) abnormal C&C connections were established between some internal nodes and external nodes only when sophisticated supply chain attacks (e.g., 2020 SolarWinds breach) were going on.

To detect such abnormal data exfiltration and C&C traffic at the enterprise egress point, we can collect normal traffic patterns which do not contain the abnormal data exfiltration and C&C traffic that only appear when some attack is going on. Such a profile of normal traffic can be used as the baseline in the traffic anomaly detection.

Assuming all the data exfiltration and C&C traffic in the supply chain attacks are encrypted and laundered through stepping stones, we have to use some traffic characteristics that is invariant after encryption and laundering across stepping stones to identify and track those attack information flows.

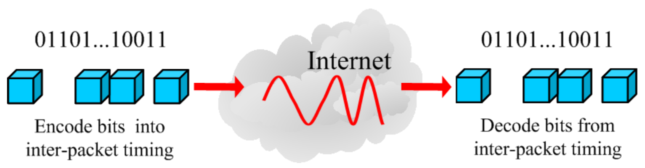
01101...10011
Internet
01101...10011
Encode bits into inter-packet timing
Decode bits from inter-packet timing

Fig. 2. Tagging Packet Flows via Encoding Unique Bit String into the Inter-Packet Timing of Packet Flows

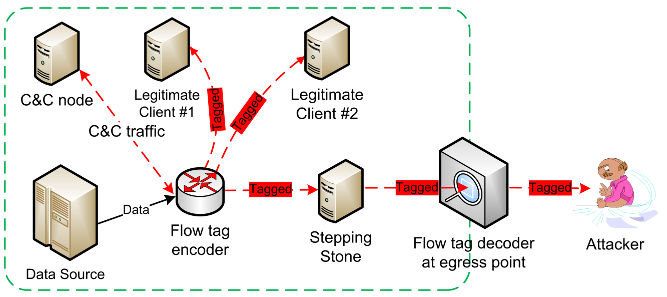

Fig. 3. Detecting and Tracking the Attack Information Flows During Supply Chain Attacks

Given any packet flow $p_1, \ldots, p_n$ $(n > 1)$, each packet p_i has its distinct arrival (or departure) time t_i. Previous research [16] has shown that the inter-packet timing of packet flow remains relatively stable across many hops of routers and intermediate stepping stones. In addition, encryption of packets has negligible impact on the inter-packet timing. Since all the routers and applications are designed to process the packets or requests as soon as possible, they will keep the timing correlation between the incoming traffic and the outgoing traffic. Therefore, there exists mutual information in the packet timing domain across all the routers and application gateways.

We propose to use the proven inter-packet timing based flow watermarking technologies [17], [18], [19], [20], [21], [22] to uniquely tag all outgoing traffic from selected data sources and mission critical systems. As shown in Figure 2, we can transparently encode an arbitrary l-bit string w (as a watermark) into the inter-packet timing of a sufficiently long packet flow by slightly adjusting the timing of selected packets. If the encoded bit string is unique and robust enough, such bit encoding essentially tags the packet flow in the inter-packet timing domain. Since the information encoding is in the inter-packet timing which neither changes nor depends on the packet content, it is applicable to any packet flow of any protocol even if the packet flow is encrypted. By transparently encoding unique bit string into the inter-packet timing of the given packet flow, one can reliably identify and track the tagged packet flow across stepping stones by checking if any given flow contains the encoded bit string.

Figure 3 illustrates how we can detect and track the attack information flows during the supply chain attack. One key component is the flow tag encoder, which is a special router that transparently encodes the given watermark into the inter-packet timing of specified (potentially encrypted) packet flows in real-time. We can deploy one such flow watermark encoder in front of each protected data source or mission critical system such that all the packet flows to or from the data source or mission critical system will pass the watermark encoder. This enables us to tag each outgoing packet flow from each of the chosen data sources or mission critical systems with an unique watermark. Since the inter-packet timing based flow watermark encoder only slightly adjusts the timing of selected packets without ever changing any content of any packet, legitimate users can access the protected mission critical systems or data sources as usual without any changes.

To protect a given mission critical data source, we prohibit any direct data access from outside, and only allow legitimate data access via specific route and access point. For example, we can mandate that data access from mission critical server X must be done via some access point Y, and we can place separate flow tag encoders in front server X and access point Y such that packet flow from/to X and Y will be uniquely tagged unique bit string X_b and Y_b respectively.

To keep track of legitimate outgoing traffic, the egress point should maintain the following tuples indexed by the unique tag or bit string: $\langle tag, time, srcIP, dstIP, srcPort, dstPort, protocol \rangle$.

When some supply chain attack is exfiltrating data from a protected data source or laundering C&C traffic via some protected node, those data exfiltration, C&C traffic will be transparently tagged with unique bit string that identifies where they are tagged. At the egress points (e.g., firewall), we can check the inter-packet timing of each outgoing packet flows for the tags that we have encoded to the outgoing packet flows from all the protected data sources. For example, if the egress point sees some outgoing traffic is tagged with bit string X_b for outgoing traffic from mission critical server X, we have very high confidence that the traffic tagged with X_b is indeed from mission critical server X even if the traffic is encrypted and laundered through stepping stones. Since legitimate access of server X should be via access point Y whose traffic should be tagged with Y_b, we know for sure that the outgoing traffic with tag X_b is illegal. This allows us to detect and stop the encrypted data exfiltration, encrypted C&C communication and the ongoing supply chain attack within a few minutes.

C. *Security and Robustness Analysis*

Our information flow based attack detection and tracking are built upon transparent flow watermarking in the inter-packet timing domain. To protect mission critical servers within an enterprise network, we need to deploy one flow tag encoder, as a special router, in front of each server to be protected, and deploy one flow tag decoder at each egress point of the network. The flow tag encoder and the flow tag decoder are completely independent from the servers to be protected. Therefore, nothing needs to be changed or installed in the servers to be protected, and the flow tag encoder and decoder still function as expected when the servers are somehow compromised (e.g., via supply chain attack).

After we have encoded $l > 0$ bits string w into the inter-packet timing of a packet flow, let w' be the l bits decoded from the tagged packet flow at the receiver side. We use $H(w, w')$ to represent the Hamming distance between w and w' which measures number of mismatched bits caused by the network distortion to the inter-packet timing of the packet flow. Instead of requiring $H(w, w') = 0$, we consider the packet flow has tag w if the Hamming distance between the decoded tag w' and w $H(w, w') < h$ where $0 \leq h < l$ is the Hamming distance threshold. Let $0 < p < 1$ be the probability that each decoded bit matches the encoded bit, then the expected l bits decoding true positive rate with Hamming distance threshold h is

$$\text{TPR}(l, h, p) = \sum_{i=0}^{h} \begin{pmatrix} l \\ i \end{pmatrix} p^{l-i}(1-p)^{i} \quad (1)$$

We use decode collision to denote the unlikely situation that an untagged packet flow happens to have the l bits decoded w' such that $H(w, w') \leq h$. Assuming the l-bits w' decoded from a random flow is uniformly distributed, then the expected decode collision probability between a random untagged packet flow and any particular l bits w is

$$\text{FPR}(l, h) = \sum_{i=0}^{h} \begin{pmatrix} l \\ i \end{pmatrix} (\frac{1}{2})^{l} \quad (2)$$

Therefore, we can have very low collision probability 5.65×10^{-5} by using $l = 32$ bits while allowing $h = 5$ mismatched bits. Previous research [23] have shown that we only need less than 60 seconds and a couple of hundred packets to achieve virtually 100% decoding true positive rate. Given the enormous amount of the data exfiltrated (e.g., tens of millions of records) in supply chain attacks, our proposed information flow based detection will have more than enough packets to achieve both high true positive rate and exceedingly low false positive rate at the same time.

V. Discussion

Built upon novel combination of packet flow watermarking and traffic anomaly detection, our proposed attack information flow based detection has the following desirable features:

1) it does not depend on any attack signature, and it can be effective against previously unknown supply chain attacks.
2) it does not impact the normal access and use of the to be protected servers.
3) it does not introduce any performance overhead nor any change to the to be protected servers.
4) it works with any packet based protocol such as IPv4, IPv6, HTTP, SSL, SSH.
5) it can be deployed incrementally and it can work seamlessly with existing IT infrastructure.

When the egress flow tag decoder has been trained properly with normal traffic, the proposed detection system can reliably detect novel supply chain attacks that involve previously unseen data exfiltration and/or C&C traffic in near real-time without detection false positive. On the other hand, the proposed detection system may miss certain software supply chain attacks such as Heartbleed.

We have not empirically validated the proposed detection system due to lack of access to the exploit code of software supply chain attacks.

VI. Related Works

To the best of our knowledge, there is no published work on detecting software supply chain attacks.

Randomization based defense Many cyber defense approaches [24], [25], [26], [27], [28], [29], [30], [31] have been proposed based on the idea of randomization. By randomizing the run-time memory address of stack, heap etc., address space layout randomization [24], [27], [28], [31] make it hard for the attack code to access the correct address at run-time. The Instruction set randomization [25], [26] makes it infeasible for the injected attack code to execute as expected. System call randomization [29] causes the attack code to use the wrong system call by randomizing the system call number at run-time.

System call based defense Many approaches have been proposed to detect cyberattacks based on checking system call sequence [32], [33], [34], [35] or tagged system call [36].

Control flow integrity (CFI) based defense Control flow integrity (CFI) [37], [30], [38], [39], [40] aims to prevent, detect control flow hijacking and code-reuse attacks by ensuring that the run-time indirect control flow transfers can only reach legitimate destinations in the control flow graph (CFG).

Unfortunately, software supply chain attack can defeat all existing randomization based, system call based and CFI based defenses by simply planting a backdoor trojan in the software update.

A number of approaches (e.g., [41], [42], [18], [19], [20], [21]) have been proposed to transparently encode bits into the inter-packet timing of a given packet flow. Our proposed attack information flow based detection uses such packet flow watermarking approaches as a building block.

VII. Conclusions

In this work, we have examined the problem of supply chain attacks and the technical challenges in detecting software supply chain attacks. We show that by novel combination of traffic anomaly detection and packet flow tagging techniques it is indeed feasible to detect certain software supply chain attacks in near real-time without any prior knowledge of the attack.

Our analysis shows that the proposed attack information flow based detection can be highly accurate with very low false positive in production environment. It is an area of future work to empirically validate the proposed detection with real world supply chain attacks.

References

[1] K. Thompson, "Reflections on Trusting Trust," *Communications of the ACM*, vol. 27, no. 8, Augeust 1984.

[2] B. Krebs, "Target Hackers Broke in Via HVAC Company," February 2014, https://krebsonsecurity.com/2014/02/target-hackers-broke-in-via-hvac-company/.

[3] I. Security, "Cost of a Data Breach Report 2019," 2019, https://www.all-about-security.de/fileadmin/micropages/Fachartikel_28/2019_Cost_of_a_Data_Breach_Report_final.pdf.

[4] "2020 United States federal government data breach," https://en.wikipedia.org/wiki/2020_United_States_federal_government_data_breach.

[5] L. Abrams, "SolarLeaks site claims to sell data stolen in SolarWinds attacks," January 2021, https://www.bleepingcomputer.com/news/security/solarleaks-site-claims-to-sell-data-stolen-in-solarwinds-attacks/.

[6] L. Tung, "Microsoft: Solarwinds attack took more than 1,000 engineers to create."

[7] B. D. Williams, "Revealed: Secret FBI Cyber Op To Clean Exchange Servers," April 2021, https://breakingdefense.com/2021/04/doj-reveals-secret-fbi-op-to-clean-exchange-servers/.

[8] A. Press, "A 'Colossal' Ransomware Attack Hits Hundreds Of U.S. Companies, A Security Firm Says," July 2021, https://www.npr.org/2021/07/03/1012849198/ransomware-cyber-attack-revil-attack-huntress-labs.

[9] C. Timberg and E. Nakashima, "The U.S. government spent billions on a system for detecting hacks. The Russians outsmarted it," February 2021, https://www.seattletimes.com/nation-world/the-u-s-government-spent-billions-on-a-system-for-detecting-hacks-the-russians-outsmarted-it/.

[10] L. Tung, "Supply chain attacks are getting worse, and you are not ready for them," August 2021, https://www.zdnet.com/article/supply-chain-attacks-are-getting-worse-and-you-are-not-ready-for-them/.

[11] E. U. A. for Cybersecurity, "ENISA Threat Landscape For Supply Chain Attacks," July 2021, https://www.enisa.europa.eu/publications/threat-landscape-for-supply-chain-attacks.

[12] FireEye, "Highly Evasive Attacker Leverages SolarWinds Supply Chain to Compromise Multiple Global Victims With SUNBURST Backdoor," December 2020, https://www.fireeye.com/blog/threat-research/2020/12/evasive-attacker-leverages-solarwinds-supply-chain-compromises-with-sunburst-backdoor.html.

[13] R. Staff, "SolarWinds hack was 'largest and most sophisticated attack' ever: Microsoft president," February 2021, https://www.reuters.com/article/us-cyber-solarwinds-microsoft/solarwinds-hack-was-largest-and-most-sophisticated-attack-ever-microsoft-president-idUSKBN2AF03R.

[14] S. Chazelas, "GNU Bash CVE-2014-6271 Remote Code Execution Vulnerability," September 2014, https://www.securityfocus.com/bid/70103.

[15] Symantec, "Internet Security Threat Report Volume 24," February 2019, https://www.symantec.com/content/dam/symantec/docs/reports/istr-24-2019-en.pdf.

[16] X. Wang, D. S. Reeves, and S. F. Wu, "Inter-Packet Delay based Correlation for Tracing Encrypted Connnections through Stepping Stones," in *Proceedings of the 7th European Symposium on Research in Computer Security (ESORICS 2002)*, ser. LNCS-2502. Springer-Verlag, October 2002, pp. 244–263.

[17] A. Iacovazzi and Y. Elovici, "Network Flow Watermarking: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, pp. 512 – 530, First Quarter 2017.

[18] X. Wang, S. Chen, and S. Jajodia, "Tracking Anonymous Peer-to-Peer VoIP Calls on the Internet," in *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS 2005)*. Alexandra, VA: ACM, November 2005, pp. 81–91.

[19] Y. J. Pyun, Y. H. Park, X. Wang, D. S. Reeves, and P. Ning, "Tracing Traffic through Intermediate Hosts that Repacketize Flows," in *Proceedings of the 26th Annual IEEE Conference on Computer Communications (Infocom 2007)*, May 2007.

[20] W. Yu, X. Fu, S. Graham, D. Xuan, and W. Zhao, "DSSS-Based Flow Marking Technique for Invisible Traceback," in *Proceedings of the 2007 IEEE Symposium on Security and Privacy (S&P 2007)*. IEEE, May 2007.

[21] X. Wang, S. Chen, and S. Jajodia, "Network Flow Watermarking Attack on Low-Latency Anonymous Communication Systems," in *Proceedings of the 2007 IEEE Symposium on Security & Privacy (S&P 2007)*, Oakland, CA, May 2007, pp. 116–130.

[22] Z. Ling, J. Luo, D. Xu, M. Yang, and X. Fu, "Novel and Practical SDN-based Traceback Technique for Malicious Traffic over Anonymous Networks," in *Proceedings of the 38th Annual IEEE Conference on Computer Communications (Infocom 2019)*, May 2019.

[23] X. Wang, "On the Feasibility of Real-Time Cyber Attack Attribution on the Internet," in *Proceedings of the 35th IEEE MILCOM*, November 2016.

[24] S. Bhatkar, D. C. DuVarney, and R. Sekar, "Address Obfuscation: An Efficient Approach to Combat a Broad Range of Memory Error Exploits," in *Proceedings of the 12th USENIX Security Symposium*, August 2003.

[25] G. S. Kc, A. D. Keromytis, and V. Prevelakis, "Countering Code-Injection Attacks with Instruction-Set Randomization," in *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS 2003)*. ACM, October 2003, pp. 272–280.

[26] E. Barrantes, D. Ackley, S. Forrest, T. Palmer, D. Stefanovic, and D. Zovi, "Randomized Instruction Set Emulation to Disrupt Binary Code Injection Attacks," in *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS 2003)*. ACM, October 2003, pp. 281–289.

[27] J. Xu, "Intrusion Prevention Using Control Data Randomization," in *Proceedings of the 2003 International Conference on Dependable Systems and Networks (DSN 2003)*. IEEE, June 2003.

[28] Z. K. Jun Xu and R. K. Iyer, "Transparent Runtime Randomization for Security," in *Proceedings of the 22nd Symposium on Reliable and Distributed Systems (SRDS 2003)*. IEEE, October 2003.

[29] X. Jiang, H. J. Wang, D. Xu, and Y.-M. Wang, "RandSys: Thwarting Code Injection Attacks with System Service Interface Randomization," in *Proceedings of the 26th Symposium on Reliable and Distributed Systems (SRDS 2007)*. IEEE, October 2007.

[30] C. Zhang, T. Wei, Z. Chen, L. Duan, L. Szekeres, S. A. Mccamant, D. Song, and W. Zou, "Practical Control Flow Integrity & Randomization for Binary Executables," in *Proceedings of the 2013 IEEE Symposium on Security and Privacy (S&P 2013)*. IEEE, May 2013.

[31] S. J. Crane, S. Volckaert, F. Schuster, C. Liebchen, P. Larsen, L. Davi, A.-R. Sadeghi, T. Holz, B. D. Sutter, and M. Franz, "It's a TRaP: Table Randomization and Protection against Function-Reuse Attacks," in *Proceedings of the 22nd ACM Conference on Computer and Communications Security (CCS 2015)*. ACM, October 2015, pp. 243–255.

[32] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, "A Sense of Self for Unix Processes," in *Proceedings of the 1996 IEEE Symposium on Security and Privacy (S&P 1996)*. IEEE, May 1996.

[33] C. Warrender, S. Forrest, and B. Pearlmutter, "Detecting Intrusions Using System Calls: Alternative Data Models," in *Proceedings of the 1999 IEEE Symposium on Security and Privacy (S&P 1999)*. IEEE, May 1999, pp. 133–145.

[34] R. Sekar, M. Bendre, and P. Bollineni, "A Fast Automaton-Based Method for Detecting Anomalous Program Behaviors," in *Proceedings of the 2001 IEEE Symposium on Security and Privacy (S&P 2001)*. IEEE, May 2001.

[35] H. H. Feng, O. M. Kolesnikov, P. Fogla, W. Lee, and W. Gong, "Anomaly Detection Using Call Stack Information," in *Proceedings of the 2003 IEEE Symposium on Security and Privacy (S&P 2003)*. IEEE, May 2003.

[36] X. Wang and X. Jiang, "Artificial Malware Immunization based on Dynamically Assigned Sense of Self," in *Proceedings of the 13th Information Security Conference (ISC 2010)*, October 2010.

[37] M. Abadi, M. Budiu, U. Erlingsson, and J. Ligatti, "Control-Flow Integrity: Principles, Implementations, and Applications," in *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS 2005)*. ACM, November 2005, pp. 340–353.

[38] M. Zhang and R. Sekar, "Control Flow Integrity for COTS Binaries," in *Proceedings of the 22nd USENIX Security Symposium*. USENIX, August 2013.

[39] V. Mohan, P. Larsen, S. Brunthaler, K. W. Hamlen, and M. Franz, "Opaque Control-Flow Integrity," in *Proceedings of the 22th Network and Distributed System Security Symposium (NDSS 2015)*, February 2015.

[40] V. van der Veen, E. Göktas, M. Contag, A. Pawlowski, X. Chen, S. Rawat, H. Bos, T. Holz, E. Athanasopoulos, and C. Giuffrida, "A Tough Call: Mitigating Advanced Code-Reuse Attacks at the Binary Level," in *Proceedings of the 2016 IEEE Symposium on Security and Privacy (S&P 2016)*. IEEE, May 2016.

[41] X. Wang and D. S. Reeves, "Robust Correlation of Encrypted Attack Traffic through Stepping Stones by Manipulating of Interpackets Delays," in *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS 2003)*. ACM, October 2003, pp. 20–29.

[42] P. Peng, P. Ning, D. S. Reeves, and X. Wang, "Active Timing Based Correlation of Perturbed Traffic Flow with Chaff," in *Proceedings of the 2nd International Workshop on Security in Distributed Computing Systems (SDCS-2005)*, June 2005.