Toward Mitigation-as-a-Service in Cooperative Network Defenses

Stephan Mannhart, Bruno Rodrigues, Eder Scheid, Salil S. Kanhere*, Burkhard Stiller
Communication Systems Group CSG, Department of Informatics IfI, University of Zurich UZH, Switzerland
E-mail: stephan.mannhart@uzh.ch, [rodrigues,scheid,stiller]@ifi.uzh.ch
**Networked Systems and Security Group NetSyS, UNSW Sydney, NSW 2052 Australia*
E-mail: salil.kanhere@unsw.edu.au

***Abstract*—Distributed Denial-of-Service (DDoS) attacks are by design highly decentralized and therefore hard to defend against. By utilizing a decentralized, multi-domain, cooperative defense mechanism, it is possible to combine software and hardware capabilities to effortlessly mitigate large scale attacks. Cooperative defense systems face many challenges, such as deployment complexity due to high coordination overhead, reliance on trusted and stable channels for communication and the need for effective incentives to bolster cooperation among all involved parties. In particular, incentives are the key to ensure successful deployment of a "Mitigation-as-a-Service (MaaS)" for cooperative defense systems. This paper discusses the critical issue of providing a proof of the effectiveness of a cooperative defense mitigation, considering four state-of-the-art solutions toward an independently verifiable proof of mitigation. A qualitative analysis of these approaches across 9 dimensions shows that none satisfy all requirements due to the inherent trade-offs between practicability and security. As a result, it is identified that the issue of authenticating the underlying network flows remains unsolved.**

I. Introduction

The growing threat of Distributed Denial-of-Service (DDoS) attacks requires novel, fast-acting countermeasures. Mitigating large-scale DDoS attacks at the attack-traffic target is infeasible due to the overwhelming bandwidth of these attacks. A mechanism that allows multiple Autonomous Systems (AS) to mitigate the attack near its source within the managed address space of each AS can be an effective solution. For instance, it allows to combine the detection/mitigation capabilities of the cooperative entities, reduce the detection/mitigation overhead in a single entity, and block malicious traffic near its source. Enterprises targeted by DDoS attacks could subscribe to such a cooperative defense service by paying a fee to be protected from future threats. However, cooperative defense systems face three main challenges as outlined in [12], which include deployment complexity due to high coordination overhead, reliance on a trusted and stable channel for communication and the effective use of incentives to bolster cooperation among involved parties.

Incentives are necessary to cover CAPital EXpenditures (CAPEX) to set up communication infrastructures including additional hardware and software acquisition costs, and OPErating EXpenditures (OPEX) are incurred as soon as a mitigation service is in use. A possible solution to cover these expenditures is by passing them on to the service customer. Service fees paid by the customer can then be used as incentives among ASes involved in the cooperative defense to motivate the collaborative behavior.

However, the use of complex incentive schemes would lead to even higher complexity of operation, corresponding to an increase in subscription fees to cover OPEX and CAPEX. Incentive payout therefore needs to be based on an automated mitigation mechanism that can be verified independently to simplify and streamline the entire process. Additional costs related to delays and negotiations between ASes to verify whether the service was performed as agreed could then be avoided.

Avoidable costs include additional delays and negotiations between ASes after a mitigation has been conducted to verify whether the service was performed as agreed. However, since the effectiveness of a mitigation task needs to be manually verified by the target AS by examining logs and changes in attack traffic, an independently verifiable and automated mitigation proof is essential for the incentive scheme. Hence, automatic checks of the effectiveness of a mitigation, and subsequently, automatic payouts of incentives between ASes could be performed.

This paper discusses possible approaches to provide such an independently verifiable mitigation-proof and the challenges associated with each approach. We show that no satisfiable solution can be provided since a trade-off exists between practicability and security of each approach. In this regard, most of the discussed approaches are not clearly separable since they combine similar concepts and even a possible combination of these approaches would lead to an increase in complexity and lack of scalability. In addition to that, all of the approaches fail to include the integrity of the underlying network flows which leaves room for tampering and falsification of the mitigation proof. The literature research performed did not indicate other approaches which might address differently the core challenges as shown with the approaches presented in this paper.

This paper is organized as follows: Section II describes the relevance of a mitigation proof in the context of a cooperative defense. Section III presents approaches toward an automated solution, and in Section IV these approaches are compared. Section V concludes the work.

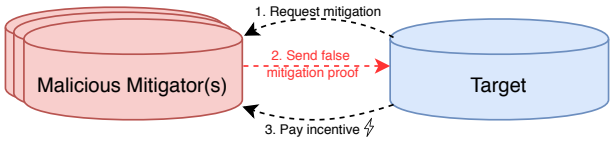

Figure 1. False-reporting of mitigation proof in a cooperative defense

II. Mitigation in a Cooperative Defense

The underlying process of a DDoS mitigation service involves multiple ASes collaboratively countering the DDoS attack. As soon as an attack is detected at an AS T (Target), it contacts ASes participating in the alliance to request mitigation. Then, an AS M (Mitigator), managing the address range responsible for the attack has the choice of accepting the mitigation request and providing the service by filtering the attack traffic.

A proof-of-mitigation (*cf.*, Figure 1) needs to exist to convey service completion to AS T in a way that will clearly confirm that a cooperative attack request has been successfully mitigated. Thus, upon acknowledgment of this proof, an incentive can directly be paid by AS T to AS M, which completes the process.

The crucial element of this exchange is the ability of an AS M to provide a proof-of-mitigation that satisfies the aspects of reproducibility, tamper-evidence and timeliness. If such a proof is not available right after completion of the mitigation, the other party will withhold the incentive payout and the overall service becomes unusable. Manual verification of a mitigation proof is also not feasible, due to the strict time constraints required to provide a fast-acting mitigation service able to counteract large-scale DDoS attacks. The timeliness aspect does therefore also include the aspect of being able to automatically verify the proof during the available time-window and excludes any user interaction in order to be efficient.

If the mitigation proof is falsified, the process fails, and the mitigation service cannot be provided, or the target AS erroneously pays the incentive since it failed to realize the falsification of the mitigation proof. Both scenarios would lead to a breakdown of the mitigation service offering since an inherent trust between both parties would be required instead of being able to rely on a verifiable proof.

For a qualitative discussion of the individual approaches presented in this paper, metrics are based on the scheme proposed by Zargar *et al.* [12] and focus on the deployment complexity as well as scalability while adding security related metrics not present in [12].

These additional metrics can provide an important overview of the presented approaches since providing a successful proof of mitigation is largely dependent on the security of the system generating the proof as well as the proof itself.

1) **Confidentiality**: Describes how effective rules and measures are to protect both the mitigation proof as well as the mitigation system from unauthorized access.
2) **Integrity**: Defines the level of trustworthiness of the proof generated by the given approach in regards to accuracy and tamper-resistance.
3) **Availability**: Relates to the availability of the mitigation proof to authorized parties.
4) **Reproducibility**: Describes the ability to reproduce the proof through replay by a third party.
5) **Tamper-Evidence**: Discloses the effort necessary for changing the proof of mitigation to reflect an alternate reality.
6) **Timeliness**: Defines the ability to provide an automatically verifiable proof within a pre-specified time-frame while adhering to all security requirements.
7) **Deployment Complexity**: Defines the additional resources required to deploy the approach.
8) **Scalability**: Relates to the adaptability of the approach in regards to a large-scale DDoS defense scenario with high-bandwidth attacks.
9) **Service Model**: Defines the cloud service model the approach most closely relates to.

III. Approaches Toward Mitigation Proofs

This Section presents approaches toward MaaS providing a short description overviewing their functioning and discussing their advantages and disadvantages.

A. Marketplace of Mitigation VNFs

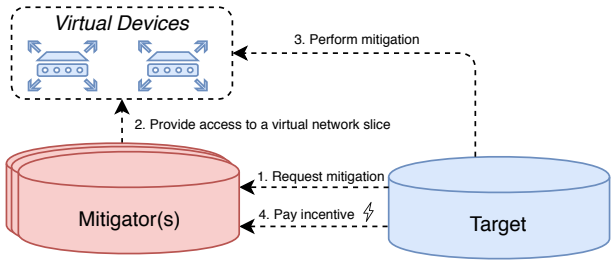

Figure 2. Mitigation device based on VNF available in a trusted marketplace

*1) **Description**:* Network Function Virtualization (NFV) [4] can provide an efficient solution to the deployment conundrum by allowing the mitigation system to be encapsulated as Virtualized Network Functions (VNF), which can directly be deployed on commodity hardware running on-site at the AS. Thus, to support a simple deployment of this approach, a VNF "marketplace" could be built as a platform of VNFs for all ASes involved in the cooperative defense. The mitigator AS would load the VNF directly from the marketplace to provide the mitigation service. This ensures that the VNF image used to conduct the mitigation is known

by all ASes and its integrity can be checked by comparing a hashed checksum of the image to a stored value on the marketplace. The marketplace also allows logging of access to the VNFs by the individual AS. Local caching of VNFs can be used to avoid increased load on the VNF marketplace.

*2) **Advantages**:* The high degree of isolation of this approach is a clear advantage. VNFs contain the minimal code necessary to conduct the mitigation and can therefore directly contribute to a certain degree of trust by implicitly ensuring that the correct code is executed. Running the mitigation inside VNFs also eases the deployment for ASes interested in offering such a service since they can run the VNF without any additional setup on supported hardware. Running VNFs only requires the necessary infrastructure for data plane control and can even be accomplished without directly using Software-Defined Networking (SDN) as outlined in [4].

*3) **Disadvantages**:* Encapsulating the mitigation service inside VNFs does not guarantee that the AS mitigator will not be able to tamper with the VNF itself. This means that the act of deploying the mitigation service through VNFs alone is not a reliable proof of mitigation and the AS requesting the service still needs to trust that the mitigator will only run untampered VNFs directly from the VNF marketplace. Even if the VNF has not been tampered with, network flows on which the mitigation will be conducted could be manipulated before they reach the VNF which could lead to a minimized or non-existent workload for the mitigator while still receiving the incentive payout from the AS requesting the service.

B. Trusted Computing

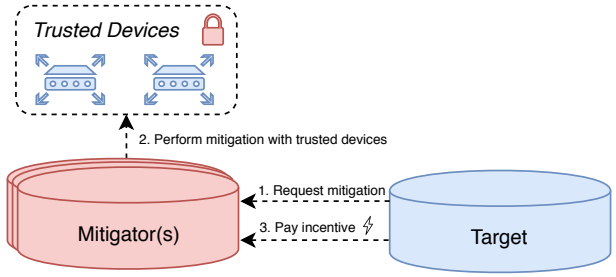

Figure 3. Mitigation devices are based on Trusted Platform Modules (TPM)

*1) **Description**:* A Trusted Platform Module (TPM) allows the secure storage of verification hashes in the on-chip Platform Configuration Registers (PCR) that can be used to attest a secure boot chain from BIOS over Bootloader, OS up to the Hypervisor [6]. Thus, code running inside the VM can be partly isolated by using technologies such as Intel Software Guard Extension (Intel SGX) which provides a secure enclave to store application states as proposed by Shih *et al*. [11].

To further extend the chain of trust up to the VNF itself, approaches like ”vTPM” [7] aim to fill this gap with a virtualized TPM instance. However, by virtualizing the hardware-based trust, a trade-off arises where the end-system gains a level of additional protection but is still not as secure as it could be with full hardware-based trusted computing. Another solution is remote attestation, where platform- as well as VNF-states are compared with known good values by a remote system. Traditionally, remote attestation was limited to non-virtualized systems and would therefore not directly apply to a VNF-based approach.

Ravidas *et al*. [9] propose a remote attestation server that goes beyond the platform-only integrity check and allows for full attestation of VNFs through image integrity checks. This is accomplished by introducing an external Trusted Security Orchestrator (TSecO) [9] which will receive VNF launch requests from a modified Virtual Infrastructure Manager (VIM). The decision to allow a launch request can then be based on whether the checksum for the VNF is featured in a whitelist published on the marketplace. These VNF image integrity checksums can later directly be used to check against known good values for remote attestation.

*2) **Advantages**:* The full chain of trust from hardware to the VNF guarantees that only the code that is approved by the cooperative defense can be run on the designated system. This guarantee combined with the marketplace that transparently provides the actual mitigation VNF creates a cooperative defense in which mitigation requests are always handled by known and trusted VNFs without the need of trusting the operator responsible for the AS providing the mitigation service.

*3) **Disadvantages**:* The biggest drawback of using a trusted computing approach to guarantee the integrity of the mitigation system are the strict hardware requirements. The TPM is a feature available only as a standalone chip or as a solution integrated into the motherboard but it does not come pre-installed on all systems, which greatly limits the potential of a DDoS mitigation system based on this technology. The same is true for Intel SGX, which is directly integrated into many Intel CPUs but not available for all Intel CPUs and is unavailable on competing products like AMDs CPUs.

Due to the nature of a trusted computing environment, the operator of the mitigating AS also gives up full control over their system. The policies enforced by the trusted computing environment will no longer allow the operator to run any VNF on the system equipped with a TPM since only whitelisted VNFs will be allowed in order to fully enforce the chain of trust. An additional problem similar to the NFV approach without TPM is the lack of control over the underlying networking infrastructure. The mitigator could once again change the network flow to the system running the mitigation VNF and lead it to believe that it is seeing all the traffic while in reality, parts or all of the

attack traffic have been rerouted and no mitigation seems to be required anymore. This would directly prompt AS T to initiate the incentive payout since the mitigation seems to have been concluded and all trusted computing related checks were passed.

C. Secure logging

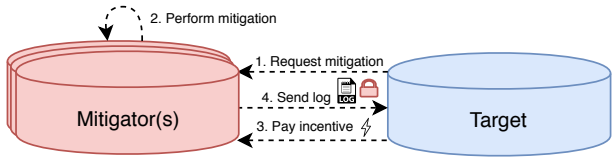

Figure 4. Secure Logging

*1) **Description**:* The main challenge in providing an independently verifiable mitigation proof is the reliance on the underlying networking infrastructure. Mitigating a DDoS attack by filtering traffic for example through blackholing requires the mitigator to have access to the networking infrastructure and to trust that the traffic represented by the infrastructure is the actual traffic passing through the mitigators AS. In the collaborative defense scenario, AS M in the role of the mitigator has full control over their underlying networking infrastructure which allows them to generate a proof of mitigation from the available traffic data.

This proof could consist of a detailed network log showing the effect of the mitigation as a reduction in the attack traffic from the attack source to the DDoS target. To secure this log, a scheme as presented in [10] could be used which utilizes authentication keys for the entire log as well as individual log entries together with a hash-chain to enable discovery of tampering attempts. This kind of secure logs have successfully been used to build various cloud-based systems with a high degree of accountability, such as [5] where secure logs from a medical device have been stored while maintaining tamper evidence. To further secure this log, the trusted computing approach discussed in Section III-B can be leveraged by utilizing both TPM as well as Intel SGX which has also been demonstrated by [5]. Intel SGX has specifically been proposed as a good extension for NFV integrity by [11] since it allows to protect specific application states inside the secure enclave provided by Intel SGX. In our case, storing the traffic logs inside that enclave would add a layer of security on top of the cryptographic methods proposed by Schneier *et al.* [10].

Since the log file has been created on an isolated system, checking the integrity of the log through a third party becomes an important aspect of its overall credibility. Haeberlen *et al.* [3] proposed a remote audit to ensure the correct operation of a remote system. This works similar to remote attestation as outlined in Section III-B1 but focuses on deterministic log files instead of binaries running on a system. By remotely replaying system executions and comparing the resulting log files to the logs obtained from the remote system, the correct behavior of the system under test can be determined.

*2) **Advantages**:* Reducing the mitigation proof to the log files minimizes the complexity of the overall proof. Checking the correctness of the log suffices to establish the success of the mitigation conducted by AS M. Together with the trusted computing schemes discussed in Section III-B1, secure logging requires no additional trust in the mitigation AS and the remote audit helps to cover any trust issues presented by the logging process itself.

*3) **Disadvantages**:* Protecting log files from tampering requires that they be stored securely, for example in tamper-proof hardware such as Intel SGX enclaves. This presents a similar disadvantage as with the trusted computing approach due to the requirement of specialized hardware. The secure enclave itself also needs to be protected by a chain of trust including all stages of the boot process right up to the operating system. To enable this, a TPM needs to be present in the system, which limits the hardware choice for a mitigation system even further.

Also, capturing the mitigation of a high-volume DDoS attack through network traffic results in large log files. Transferring these log files for remote auditing or to be used as the mitigation proof introduces additional delays in the mitigation process due to their large size. Storing logs to keep track of past mitigations also becomes a burden due to the immense disk space requirements.

D. Network Slicing

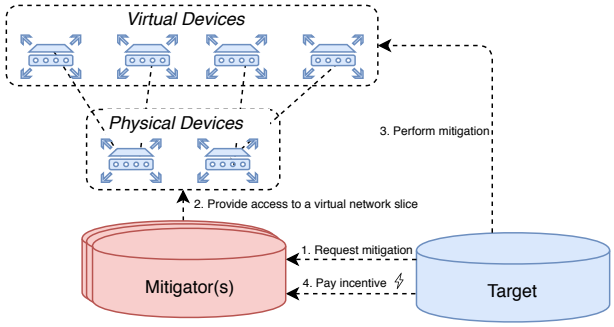

Figure 5. Virtual Slice

*1) **Description**:* Recent advances in network virtualization technologies and Software-Defined Networks (SDN) can be leveraged to realize network slicing as a service [13] for mitigation purposes. By requesting mitigation services from AS M, the target AS T gains access to a virtualized network slice. Based on the IP addresses provided with the mitigation request, the slice can be configured to only provide access to observe the flows of the attacking IP addresses.

Slice generation could be accomplished with "AutoSlice" proposed by Bozakov *et al*. [1] which creates on-demand virtualized SDN networks (vSDN)[1]. These vSDNs can be controlled by standard SDN controllers which would allow AS T to directly perform the mitigation on the target system.

*2) **Advantages**:* This approach shifts the requirements of the mitigation proof. There is no need for providing a direct proof that the mitigation has been carried out, since AS T has complete control over the whole process. The burden of providing the proof is lifted from AS M and the payout of the incentive from AS T can directly occur after the mitigation has been performed.

Creation of slices can be directly coupled to the mitigation request since the required information about relevant IP addresses to observe and control is directly provided together with the mitigation request.

*3) **Disadvantages**:* AS M has to give up full control of their networking infrastructure by providing the network slice as a service and AS T has to trust that the slice represents the portion of infrastructure of interest.

The implementation would also have a strong hardware as well as software requirement if realized with the AutoSlice[1] architecture: The networking infrastructure needs to be SDN-based and additional commodity servers with Open vSwitch installations are required to circumvent the constraint of limited flow-table sizes presented by most OpenFlow hardware switches[1].

IV. Discussion

Table I
Qualitative comparison of approaches toward MaaS

	NFV	Trusted Computing	Secure Logging	Network Slicing
Security				
1. Confidentiality	Low	Medium	Low	Low
2. Integrity	Low	High	High	Low
3. Availability	High	Medium	High	Medium
4. Reproducibility	High	High	Low	High
5. Tamper-Evidence	Low	Medium	Medium	Low
6. Timeliness	High	High	Low	Medium
Practicability				
7. Deployment Complexity	Low	High	High	High
8. Scalability	High	Low	Low	Low
Scope				
9. Service Model	SaaS	SaaS	PaaS	IaaS

Table I shows that no single approach satisfactorily addresses the trade-offs between security and practicability and could, therefore, be used by itself for an independent trustless mitigation service. NFV and Network Slicing have similar characteristics with respect to security due to their virtualized nature. While NFV virtualizes a single function in the network, Network Slicing aims to deliver a portion of the network infrastructure as a service. To accomplish this, approaches like AutoSlice [1] help to automate the creation of on-demand slices per mitigation request. However, the infrastructure requirements such as the need for SDN based networking for the automatic creation of vSDNs, increases the associated deployment complexity, thus limiting the applicability of this approach. An NFV-based approach on the other hand, presents a lower deployment complexity. For example, CoFence [8] can create VNFs upon demand to filter attack traffic, however, security aspects of this approach can be easily tampered with to obtain the incentive related to the mitigation service.

The Trusted Computing and Secure Logging approaches have high deployment complexity as strict hardware requirements need to be considered. Logging is by default provided by any mitigation tool and therefore has no deployment complexity, but secure logging requires a trusted platform to ensure that the output of a mitigation action has not been tampered with. These deployment complexities directly translate to poor scalability since a large number of TPM as well as Intel SGX [2] enabled systems would be required to use a trusted computing approach at scale.

The service model metric differentiates individual approaches by correlating them with their respective cloud service models. Table I presents this metric showing NFV and Trusted Computing approaches follow a similar model as Software-as-a-Service (SaaS) since these are individual software packages that provide the proof. In contrast, secure logging only provides logs identical to Platform-as-a-Service (PaaS) cloud models where an interface to a service is provided. The approach with the highest degree of access to the mitigation system is the network slicing approach where, similar to Infrastructure-as-a-Service (IaaS), a complete virtualized networking infrastructure is provided.

At the outset, it may appear that combining some of these approaches could lead to a comprehensive solution that addresses all requirements. However, as noted in Section III, there exists overlap between most of these approaches, which leads to an increase in complexity and raises scalability challenges. For instance, an NFV approach that is a low complexity approach could be combined with a high-scalability approach like Secure Logging. However, this would lead to combined drawbacks in regards to practicability which would make the resulting approach excel in regards to security compared with the individual approaches but would render it hard to deploy and scale.

The authors envision two main scenarios for future research toward a practicable MaaS offering: The first scenario entails finding a new approach of providing a mitigation proof that balances the security and practicability requirements outlined in the discussion section well enough to warrant a proof-of-concept implementation. The second strategy would introduce the assumption of a minimal degree of inherent trust among the ASes to eliminate the as of yet insurmountable task of isolating the entire mitigation infrastructure in order to prove successful mitigation. Both

strategies could quickly result in a technical demonstration of a complete MaaS architecture to foster further research in the field.

For the first scenario, additional research in the area of trusted computing might turn out to be most fruitful to find a suitable mitigation proof. An approach based on trusted computing that would undoubtedly prove that the entire infrastructure, including the underlying network flows of an AS is tamper-evident, would change the mitigation proof to be an inherent property of the system itself. This would allow for the mitigation to be automatically conducted upon receipt of the attack information while implicitly proving the mitigation service has been carried out correctly due to the trusted infrastructure it is running on.

Existing reputation algorithms can be incorporated into the collaborative defense protocol to foster cooperation and build a foundation of trust in the second scenario. Although being a more viable approach by eliminating the need for software/hardware mechanisms to verify the mitigation of an attack, its sole use does not guarantee that a malicious AS will not perform malicious actions to subvert the functioning of the reputation algorithm such as providing negative feedback to ASes that correctly execute mitigation requests or not providing feedback at all.

V. Final Considerations

Ensuring the effectiveness of a cooperative mitigation task is an important step toward the feasibility of incentives to stimulate the cooperative behavior and to cover operational costs of mitigating attacks. In this regard, this paper presents a detailed and conceptual discussion of four main approaches that could be deployed toward a verifiable MaaS. The multi-dimensional requirement analysis suggests that no single approach by itself can fully implement a trustless multi-domain cooperative defense scenario. Each approach has disadvantages concerning security or practicability and none of the approaches are able to tackle the problem of verifying the authenticity of the network flows.

References

[1] Z. Bozakov and P. Papadimitriou, "Autoslice: Automated and Scalable Slicing for Software-Defined Networks," in *Proceedings of the 2012 ACM conference on CoNEXT student workshop*. ACM, 2012, pp. 3–4.

[2] V. Costan and S. Devadas, "Intel SGX Explained." *IACR Cryptology ePrint Archive*, 2016.

[3] A. Haeberlen, P. Kouznetsov, and P. Druschel, "PeerReview: Practical Accountability for Distributed Systems," *ACM SIGOPS operating systems review*, vol. 41, no. 6, pp. 175–188, 2007.

[4] NFV White Paper, "Network Functions Virtualisation: An Introduction, Benefits, Enablers, Challenges & Call for Action. Issue 1," Oct. 2012.

[5] H. Nguyen, B. Acharya, R. Ivanov, A. Haeberlen, L. T. Phan, O. Sokolsky, J. Walker, J. Weimer, W. Hanson, and I. Lee, "Cloud-based secure logger for medical devices," in *Connected Health: Applications, Systems and Engineering Technologies (CHASE), 2016 IEEE First International Conference on*. IEEE, 2016, pp. 89–94.

[6] Nokia, "Trusted NFV Systems," Mar. 2018. [Online]. Available: https://onestore.nokia.com/asset/201400/Nokia_Trusted_NFV_Systems_White_Paper_EN.pdf

[7] R. Perez, R. Sailer, L. van Doorn *et al.*, "vTPM: Virtualizing the Trusted Platform Module," in *Proc. 15th Conf. on USENIX Security Symposium*, 2006, pp. 305–320.

[8] B. Rashidi and C. Fung, "CoFence: A Collaborative DDoS Defence Using Network Function Virtualization," in *12th International Conference on Network and Service Management (CNSM 16)*, October 2016.

[9] S. Ravidas, S. Lal, I. Oliver, and L. Hippelainen, "Incorporating Trust in NFV: Addressing the Challenges," in *Innovations in Clouds, Internet and Networks (ICIN), 2017 20th Conference on*. IEEE, 2017, pp. 87–91.

[10] B. Schneier and J. Kelsey, "Cryptographic Support for Secure Logs on Untrusted Machines," in *USENIX Security Symposium*, vol. 98, 1998, pp. 53–62.

[11] M.-W. Shih, M. Kumar, T. Kim, and A. Gavrilovska, "S-NFV: Securing NFV states by using SGX," in *Proceedings of the 2016 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*. ACM, 2016, pp. 45–48.

[12] S. T. Zargar, J. Joshi, and D. Tipper, "A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks," *IEEE Communications Surveys Tutorials*, vol. 15, no. 4, pp. pp. 2046–2069, Fourth 2013.

[13] X. Zhou, R. Li, T. Chen, and H. Zhang, "Network Slicing as a Service: Enabling Enterprises' own Software-Defined Cellular Networks," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 146–153, 2016.