A Structured Approach to Insider Threat Monitoring for Offensive Security Teams

Amir Al Sadi †, Davide Berardi †, Franco Callegati †, Andrea Melis †, Marco Prandini †, Luca Tolomei ◇
{amir.alsadi, davide.berardi, franco.callegati, a.melis, marco.prandini}@unibo.it, luca.tolomei@obsidium.it

† Department of Computer Science and Engineering University of Bologna, Bologna, Italy
◇ Obsidium S.R.L., Bologna, Italy

***Abstract*—In many countries, government agencies resort to third parties to acquire security services of many kinds, including Red Team operations to test the effectiveness of own defenses mechanisms. Absolute trust is a key requirement, lest a potentially devastating finding be exploited by a treacherous Red Team against the same entity which commissioned the operation, or sold to its adversaries. In our endeavour as a joint private-academic initiative to address this peculiar market, we observed that a structured approach to this issue is much less common than we would have expected. In this work, we outline the process we are devising to offer customers a verified environment, but integrating it with an evidence-based proof of their correct behavior during the operation, striving to solve the "Quis custodiet ipsos custodes" struggle in an offensive setting.**

***Index Terms*—Secure Infrastructure, Penetration Testing, Insider Threat, IaC**

I. Introduction

Insider threats pose a complex challenge. They represent one of the most expensive security issues for companies [1] in general. Furthermore, they have the potential to obliterate companies working in specific fields where customers' trust is the tool of the trade. A prime example is the offensive security sector, e.g. firms offering penetration testing services, which have a mission to uncover unbeknownst ways to access sensitive data or to interfere with the processes of their customers. While initial vetting of prospective employees and a sane trust relationship built on close cooperation remain fundamental, it is only prudent to deploy technological aids to detect possible malicious behaviors and to contain their effects. In this paper, we describe the initial release of an architecture for the automated deployment of a penetration testing environment including insider threat management features. The *Castrum* model is the result of a cooperation between a young start-up and a university research group to devise a system incorporating scientifically sound, state-of-the-art methodologies, at the same time taking into account real-world needs. The result is a framework for the programmable definition of the whole ecosystem of hardware components, configurations, and services needed to conduct a specific penetration testing mission in a controlled environment. The paper summarizes the related works in the area of insider threat detection in Section II, then proceeds to Section III to describe the proposed architecture for the automatic deployment of the monitoring infrastructure. Section IV presents the results of a test case, before drawing conclusions in Section V.

II. Context and related works

In a classic cybersecurity framework, e.g. NIST's, five main phases are defined: Identification, Prevention, Detection, Response and Recovery [2].
The first three phases are those related to the monitoring of the behavior of all the agents in the system. After that, the Response and Recovery phases focus on how to react to the attack once it is identified, to contain damages and restore normal operations.
Prevention would require a priori knowledge; response and recovery are next to pointless after highly sensitive data have been exfiltrated or vulnerabilities have been exploited. For these reasons we decided to focus mainly on the detection phase, implementing an efficient way to deploy a monitoring infrastructure for suspicious insider threat activities, based on a customer-oriented insider definition. We argue that this approach is more cost and resource efficient.

Literature abounds with guidelines and principles aimed at providing general descriptions of the context and the identity of the insiders [3], [4], [5]. However, experts agree that the strong contextual variance of threats [6], [7] makes providing a general yet precise identification of all possible insiders difficult.
So it's important first to build a set of rules or guidelines on how we could define the malicious behavior of an insider. In [8], for example, authors describes approaches to assemble knowledge about insider threats and to apply this knowledge in support of insider threat assessment.
Similar work has been done in [9] where authors defined a strict methodology to identify insider threats based on a better collaboration process between the information technology (IT) management and the human resources (HR) department. In [10] authors proposed a research on a comprehensive ontology of sociotechnical and organizational factors for insider threat. One of the most recent works on this same approach is [11]. This research aims to catalog human as well as technical factors associated with insider threat risks to inform the development of more proactive approaches to insider threat assessment.
In [12], the detection of the insider is based on a real-time testing simulation of real users, generating user data to test the detection of malicious users.
We found other updated and relevant resources in [13] which

is one of the latest surveys that summarized techniques for insider threat identification and detection.
What we argue is different in our approach is not the definition of an insider threat behavior but rather how we are able to detect it within a penetration testing scenario.

III. Monitoring Infrastructure

The architecture we are going to describe has two elements worth highlighting, namely: (i) the inclusion of an easily extensible set of detection tools, and (ii) the automated deployment of said tools and of the complementary data gathering and processing infrastructure. In Figure 1, the main elements of the insider threat infrastructure are depicted. Automation is of paramount importance to ensure the consistent deployment and management of all the needed components. Besides technical correctness, it also provides a guarantee that all employees' work will get profiled to detect anomalies in behavioural patterns, acting as a deterrent.
The approach to automatic configuration is based on the Infrastructure as Code (*IaC*) paradigm. As described in Section III-A, using such a technology it is possible to configure the entire infrastructure from a single centralized system.
During its life-cycle, the infrastructure is going to be continuously updated. To properly manage this process, an automated supply chain that can replicate the production environment and audit the developers changes is needed.
To ensure traceability and cope with the incremental development of the infrastructure, our proposal uses a role called *Infrastructure Source Management*. This role is in charge of maintaining the versioned source code and execute analyses on code smells. These automatized analyses and Continuous Development (CD) cycle could be reflected in test and production environment interacting with the provisioning entities, lowering the complexity of managing a flexible and non-trivial infrastructure.
The *Secret Management* role employs these measures, providing a standardized way to access data from clients, maintaining security as the first principle, and automating as many aspects as possible, to reduce error-prone manual intervention at the bare minimum required by specific processes.
Similarly, users and their roles are managed through a database, conforming to the *Identity Management* role, which is mainly used by the *Secret management* and *Infrastructure source management* roles. Therefore, the *Identity Manager*, will supervise over the access to the other roles, enabling a centralized control over the set of credentials that can be used to access the platform.
The aforementioned components functionally manage their own data-sets, producing a detailed audit trail for the *Log and analysis* role.
Whenever a log entry or combination thereof raises suspicion of malicious activity, the insider threat managers will be notified, giving them the possibility of further investigating and suggesting a response.

A. Infrastructure as Code and environment replication

To set the entire infrastructure up, we used a set of provisioning tools according to the Infrastructure as Code (IaC) paradigm. IaC entails a process based on the concept of maintaining the infrastructure description as source code, enabling the use of techniques such as versioning (e.g. with software like `git`) and code auditing.
IaC ensures continuity, as all the environments are provisioned and configured automatically, which greatly speeds up and simplifies infrastructure tests. Seamless updates positively affect security too, since legacy components gone useless and bugged software for which patches exist, are immediately taken care of. IaC also brings Continuous Integration (CI) capabilities along. CI and IaC combine to enable rapid provisioning and configuration of the environments where code is developed and tested; new tools and features can be reliably integrated into the main project trunk, following flexible delivery roadmaps such as policy checkers [14], without jeopardizing system performance and stability,

B. Traffic inspection

One of the core features of our proposed architecture is traffic analysis via deep packet inspection. This kind of monitoring could be quite expensive if centralized on a dedicated machine, which may need to employ hardware offloading or proprietary solutions to accelerate the analysis.
For this reason we chose to implement a distributed approach by default, with the possibility to switch to a centralized DPI platform. This analysis shall be aware of secure communication protocols, this will be described in details in Section IV-A.
It is important to highlight that the proposed architecture foresees specific organizational roles for each activity. An important role will be that of *insider threat manager*, the person(s) in charge, and the only authorized one(s), to receive notifications of suspicious events and to be able to access the related details.
In this paper, to provide a complete case study, we focus on a specific threat that can come from insiders, i.e. data exfiltration. To detect sensitive data spill-out, we need to identify it in the flowing traffic, which is done using the so-called *HoneyTokens* [15].
This countermeasure is obviously not a silver bullet against exfiltration, as the tokens could be identified and removed from the outgoing stream, but we argue that, in a constantly monitored environment, with a limited time, an attacker unaware of the tokens will trip onto them with high probability.

IV. Test Case Result

This infrastructure has already been tested on several real-world use cases. The goal is not to illustrate the whole process of testing a specific target, but rather to show some of the main interesting results that we are able to proof.

A. Analyzed security threats

The main threats we have analyzed are the ones we identified as preeminent for this scenarios, namely Network exfiltration

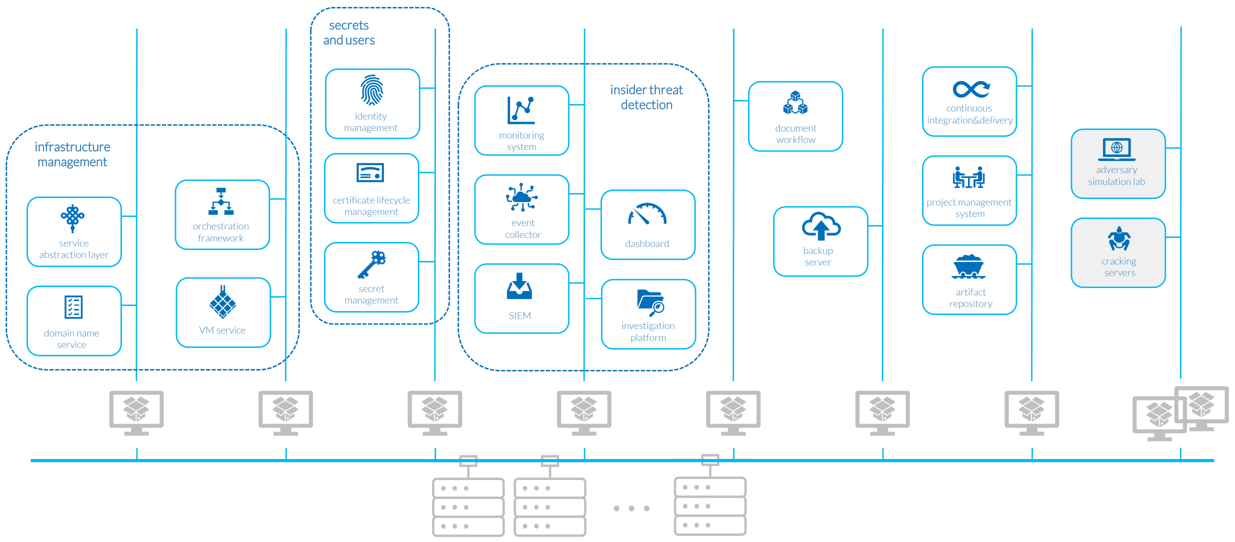
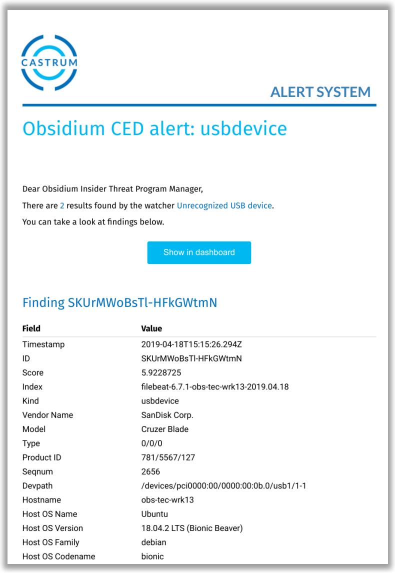


Fig. 1. Infrastructure layout with the main functions implemented by the insider threat detection platform.

and External memory exfiltration. Other methods, such as, Physical exfiltration using covert channels, described in [16] are now out of the scope of this framework relase.
As a demonstration for the possibility introduced by our analysis platform, we have evaluated Network exfiltration via secure channels, such as HTTPS.
One of the most common and reliable way to analyze HTTPS traffic is to use a MITM attack on the protocol[17], [18], without introducing weak ciphers or reducing the overall security of the protocol. To implement this probe, we had to intervene on one of the measures introduced to defeat the MITM attacks, the so-called *Certificate Key Pinning*: the browser keeps the certificate of critical sites in its internal environment to ensure that a MITM attack will not invalidate the security of the system.
Disabling this kind of support will reduce the security of the system, therefore, the implementation of the network inspection part of the platform relies on the termination of the connections using SSLsplit and the inspection of the packets on the SSLsplit server. The inspection acted by this server can be split in three parts: the first component redirects all the encrypted connections to the SSLsplit server which can therefore open it using its certificate; then, it analyzes the contents using a classic *Network intrusion detection system*, in our case Zeek; finally, SSLsplit will re-encrypt the traffic, sending it to its original destination. As stated in Section III-B, the analysis of traffic can burden a system with excessive load if it is in charge of analyzing the entire list of packets that roam the network. To optimize the network load and the work of the analysis machine furthermore, we have designed a filter for the Network intrusion detection module to search for the HoneyTokens.
When encrypted traffic is detected, it will be sent to the analysis platform as a warning. After that, regardless of the previous choice, the detection system will check if a token, pre-injected


ALERT SYSTEM

Obsidium CED alert: usbdevice

Dear Obsidium Insider Threat Program Manager,
There are 2 results found by the watcher Unrecognized USB device.
You can take a look at findings below.

Finding SKUrMWoBsTl-HFkGWtmN

Field	Value
Timestamp	2019-04-18T15:15:26.294Z
ID	SKUrMWoBsTl-HFkGWtmN
Score	5.9228725
Index	filebeat-6.7.1-obs-tec-wrk13-2019.04.18
Kind	usbdevice
Vendor Name	SanDisk Corp.
Model	Cruzer Blade
Type	0/0/0
Product ID	781/5567/127
Seqnum	2656
Devpath	/devices/pci0000:00/0000:00:0b.0/usb1/1-1
Hostname	obs-tec-wrk13
Host OS Name	Ubuntu
Host OS Version	18.04.2 LTS (Bionic Beaver)
Host OS Family	debian
Host OS Codename	bionic

Fig. 2. Alert originated by the insertion of a not allowed USB device. In this case the metadata of the analyzed machine and USB dongle are presented in the alert report.

in databases or in critical files, is present in the traffic. If the token is found, it means that specific data are being exfiltered. The External memory exfiltration detection exploits the same concepts seen for network exfiltration. Because the insertion and removal of external devices is a critical operation, it has more strict policies implemented by default in our platform. When a device is inserted or removed, the system will register the event information in a log file. For instance, in case of USB devices, the log will contain the timestamp of the insertion or the unplugging, and information on the device such Vendor and Product IDs. This log will be sent to the analysis machine.

Software	Avg. CPU	CPU max	Avg. Mem	Mem max
Zeek IDS	22.45%	100.00%	35.65%	62.81%
SSLsplit	2.83%	12.50%	0.27%	0.34%
Traffic Mirror	1.18%	1.89%	0.15%	0.34%

Fig. 3. SSL Inspection performances. The test was conducted over one hour of network downloads, brute-forces and network scans.

If this device is not present in the pre-compiled allow-list, an alarm will be triggered and sent to the responsible team.
We focused mainly on USB devices, but the procedure can be extended to every external device that has fixed information and issues events in the Linux OS: for instance, logs in the kernel buffer which are readable using dmesg.

B. Performances and Evaluations

To evaluate the usability of the method, we let different users groups perform realistic work; some of them were aware of the monitoring platform, some were not. In detail, the overall effectiveness of the platform was tested over 4 teams using 4 machines each, for a total of 12 monitored Unix machines.
We tested the system asking users to insert unregistered USB devices in the running systems and to exfiltrate a sensitive file, (`/etc/shadow`) pre-loaded with HoneyTokens, from the analyzed machines using plain-text (HTTP) and encrypted (HTTPS) connections to public websites.
In all these cases, the alerts were promptly reported to the insider threat manager to immediately analyze the anomalies. An example of the alert and the associated metadata which are sent to the email address of the insider threat manager is visible in Figure 2. In this e-mail, various metadata, useful to identify the target device and machine, are presented to the insider threat manager.
In Table 3 the performance impact of the implementation of traffic inspection described in Section III-B are shown.
This test was conducted using a VirtualBox virtual machine with 2GiB of RAM, a single virtual (uncapped) CPU and our custom Kali Linux distribution. The hypervisor was running over an Intel Core i7 CPU 870 with a nominal clock frequency of 2.93GHz. During these runs the machine under test was executing common penetration test tasks, such as network scanning using nmap, bruteforce attacks using hydra.
The results show that the system load is bearable on modern hardware; most of the processing power is required by Zeek, while SSLsplit and the inspection chain are not relevant for the system load.

V. Conclusions

Insider threats are still high on the list of dangerous attacks that companies and agencies have to deal with.
The challenges of this scenario are due to the fact that insiders have much wider freedom of access than an external attacker, so identifying what is a legitimate action or a malicious one requires much finer analyses. Create customized solutions for specific customers is one the main challenges that security service providers have to face, in order to reduce the margin for expensive mistakes, to gain the clients' trust.
Extensively exploiting virtualization technologies offered by modern operating systems, combined with the IaC paradigm, we developed a framework to deploy a complete monitoring infrastructure, customized from the client's specifications. We defined a formal process, to isolate sensitive organizational roles and to manage all the components and team members operations during a red team assignment.
We described several real-world attacks retrieved from our penetration testing campaigns. We argue that this approach might not be innovative in terms of technologies and basic concepts, but in our experience of the field it is original in its methodical application of what is needed to maximise the intended outcome such as the probability of early detection of malicious insider activities.

References

[1] V. Stavrou, M. Kandias, G. Karoulas, and D. Gritzalis, "Business process modeling for insider threat monitoring and handling," in *International Conference on Trust, Privacy and Security in Digital Business*, pp. 119–131, Springer, 2014.
[2] NIST, "Cybersecurity framework," 2020.
[3] L. Flynn, G. Porter, and C. DiFatta, "Cloud service provider methods for managing insider threats: Analysis phase 2, expanded analysis and recommendations," tech. rep., Pittsburgh University, 2014.
[4] F. Callegati, S. Giallorenzo, A. Melis, and M. Prandini, "Cloud-of-things meets mobility-as-a-service: An insider threat perspective," *Computers & Security*, vol. 74, pp. 277–295, 2018.
[5] A. Melis, M. Prandini, S. Giallorenzo, and F. Callegati, "Insider threats in emerging mobility-as-a-service scenarios," in *Proceedings of the 50th Hawaii International Conference on System Sciences*, 2017.
[6] F. Callegati, S. Giallorenzo, M. Gabbrielli, A. Melis, and M. Prandini, "Federated platooning: Insider threats and mitigations," in *Proceedings of the 52nd Hawaii International Conference on System Sciences*, 2019.
[7] B. Schneier, *Secrets and lies: digital security in a networked world.* John Wiley & Sons, 2015.
[8] F. L. Greitzer, J. Purl, Y. M. Leong, and P. J. Sticha, "Positioning your organization to respond to insider threats," *IEEE Engineering Management Review*, vol. 47, no. 2, pp. 75–83, 2019.
[9] I. H. Elifoglu, I. Abel, and Ö. Taşseven, "Minimizing insider threat risk with behavioral monitoring," *Review of business*, vol. 38, no. 2, pp. 61–73, 2018.
[10] F. Greitzer, J. Purl, D. Becker, P. Sticha, and Y. M. Leong, "Modeling expert judgments of insider threat using ontology structure: Effects of individual indicator threat value and class membership," in *Proceedings of the 52nd Hawaii International Conference on System Sciences*, 2019.
[11] F. L. Greitzer, "Insider threats: It's the human, stupid!," in *Proceedings of the Northwest Cybersecurity Symposium*, NCS '19, (New York, NY, USA), Association for Computing Machinery, 2019.
[12] S. Stolfo, "Simulated user bots: Real time testing of insider threat detection systems," 2018.
[13] I. Homoliak, F. Toffalini, J. Guarnizo, Y. Elovici, and M. Ochoa, "Insight into insiders and it: A survey of insider threat taxonomies, analysis, modeling, and countermeasures," *ACM Comput. Surv.*, vol. 52, Apr. 2019.
[14] A. Melis, D. Berardi, C. Contoli, F. Callegati, F. Esposito, and M. Prandini, "A policy checker approach for secure industrial sdn," in *2018 2nd Cyber Security in Networking Conference (CSNet)*, pp. 1–7, IEEE, 2018.
[15] F. Pouget, M. Dacier, and H. Debar, "White paper: honeypot, honeynet, honeytoken: terminological issues," *Rapport technique EURECOM*, vol. 1275, 2003.
[16] M. Guri, B. Zadov, D. Bykhovsky, and Y. Elovici, "Powerhammer: Exfiltrating data from air-gapped computers through power lines," *arXiv preprint arXiv:1804.04014*, 2018.
[17] F. Callegati, W. Cerroni, and M. Ramilli, "Man-in-the-middle attack to the https protocol," *IEEE Security & Privacy*, vol. 7, no. 1, pp. 78–81, 2009.
[18] M. Prandini, M. Ramilli, W. Cerroni, and F. Callegati, "Splitting the https stream to attack secure web connections," *IEEE Security Privacy*, vol. 8, no. 6, pp. 80–84, 2010.