14th International Defense and Homeland Security Simulation Workshop, 001
21st International Multidisciplinary Modeling & Simulation Multiconference

2724-0363 © 2024 The Authors.
doi: 10.46354/i3m.2024.dhss.001

H-Verify: Automating Intrusions through Digital Twins

Emilio Panti[1,2,*], Lorenzo Isoni[1] and Fabrizio Baiardi[1,2]

[1]Haruspex s.r.l., Largo Padre Renzo Spadoni 1, Pisa, 56125, Italy
[2]Dipartimento di Informatica, Università di Pisa, Largo B. Pontecorvo 3, Pisa, 56127, Italy

[*]Corresponding author. Email address: emilio.panti@haruspex.it

Abstract

This paper introduces H-Verify, a platform to design and implement intrusions against real-world ICT infrastructures. Unique in its approach, H-Verify leverages adversary simulations previously ran on a digital twin of the target infrastructure to fully or partially automate the planning and execution of intrusions but it can also act as a flexible decision support system for the manual planning of intrusions. Furthermore, the tool also supports the simulation results, detecting false positives in the infrastructure vulnerabilities, testing applied countermeasures, and supporting users with distinct levels of experience in red teaming engagements.

Keywords: digital twin; intrusion automation; offensive security; countermeasure validation

1. Introduction

An ICT intrusion executes a sequence of actions that includes elementary attack and information collection actions. The sequence of attacks in the intrusion results in a chain that escalate the access rights of the attackers till reaching a predefined goal, i.e. acquiring a predefined set of access rights.

H-Verify is a platform to automatically build and execute the whole sequence of actions of an intrusion either to assess the robustness of an ICT system or to penetrate it. The platform uses the outputs of the Haruspex predictive platform and an exploit database to build and execute a sequence of actions, a partial or a complete intrusion, against the target system. Its ability to support users across different proficiency levels is amplified by the availability of accurate information returned by the adversary simulations performed by the Haruspex predictive platform against the target system.

The Haruspex predictive platform runs adversary simulations against the target infrastructure. An adversary simulation is a cybersecurity assessment technique similar to the behavior of a red team that leverage its expertise to craft and execute hypothetical attack scenarios, aiming to evaluate an organization's defensive posture. The primary goal is to simulate the appearance and actions of an adversary to identify weaknesses before an actual security incident occurs. The adversary is well defined and the red team replicates the behaviors, objectives, and TTPs (tactics, techniques, and procedures) of real, known adversaries. This practice is based on comprehensive threat intelligence and seeks to replicate specific attack patterns that have been observed in the wild or are anticipated to be used against the organization. The objective is to evaluate the organization's resilience against advanced persistent threats (APTs) and to uncover realistic attack paths that could be exploited by an emulated adversary. The peculiarity of the Haruspex platform is that it automates the simulation and that it does not affect the target ICT system because it works on a digital twin of the infrastructure. As described in the following, this twin is a model of the target system.

While the Haruspex predictive platform simulates intrusion on a model of the target system, H-Verify works on the real system and it can be used either in a real intrusion or to automate the operation of read team.

When automating red team operations to improve the

target robustness, H-Verify allows to scrutinize and validate outcomes generated by the twin adversary simulations by using the exploit database to build sequences of action. This ensures a reliable validation of simulation results to discover inaccurate or incomplete information used to build the target twin, such as false positives and false negatives in the system vulnerabilities.

When used in a real intrusion, besides building and executing sequences of actions, the tool can also behave as a decision support system for the design of personalized sequences. H-Verify offers the flexibility to implement either fully or partially automated sequences to optimize efficiency and response time. During the execution, the user can interact with open sessions to exfiltrate information or upload resources, such as malware or post-exploitation tools, on compromised hosts. The exfiltrated information can be analyzed to improve the target twin and to run further simulations to discover further intrusion.

This paper is organized as follows: Section 2 introduces the Haruspex predictive platform, while Section 3 describes how it uses the digital twin technology and Monte Carlo methods for adversary simulations. Section 4 details the software architecture and the key modules of H-Verify. Experimental results from both virtual laboratory and real-world infrastructure are reported in Section 5. In Section 6, we discuss the primary limitations of our tool, which are fairly common across the broader field of offensive security applications. Section 7 reviews the main offensive tools and compares them with H-Verify. Future research directions are proposed in Section 8, and Section 9 concludes the paper with final remarks.

2. Related Work

The main innovation of H-Verify is the design of the sequences of actions in intrusions using the output of the adversary simulations by the Haruspex predictive platform. This platform builds digital twins (Husák et al., 2019; Tao et al., 2019; Lehner et al., 2021; Langlotz et al., 2022) of both the attacker (threat twin) and the target system (target twin). Each of these twin is an abstract description of an entity that focuses on those properties that are useful to discover intrusions. The target twin describes these properties for each infrastructure components while the threat twin describes the strategy, the goal, and the attack surface of an attacker. This twin also describes the possible actions of the attackers in terms of TTPs from the MITRE ATT&CK matrix (MITRE, a). These twins can be automatically generated and incorporate solutions to address any informational gaps.

Using information in these digital twins, the Haruspex platform runs adversary simulations where each one mimics one after the other the actions the attacker strategy selects and the resulting intrusions. The output of each action is determined by information in the target twin. The platform applies a Monte Carlo method (Kavak et al., 2021) that runs several independent simulations in parallel to cover alternative outputs of the actions. The final goal is the discover all the possible intrusions an attacker can build and the corresponding sequences of actions.

Hence, while the digital twins should offer an accurate information on the target infrastructure and an attacker, the adversary simulation focuses on reproducing in an accurate and complete way the interaction between the attacker and the infrastructure.

We refer to (Baiardi and Sgandurra, 2013; Baiardi et al., 2015; Baiardi, 2019; Baiardi and Tonelli, 2021; Baiardi, 2023) for full details.

3. Digital Twins for Adversary Simulations

This section briefly describes how the Haruspex predictive platform builds the digital twins and uses them in adversary simulations.

3.1. The Predictive Platform Twins

The target twin holds data about the infrastructure and the specific hardware and software modules running on each node. It provides insights about how the system is connected and how data flows within it, by modeling both the physical and logical topology of the system, including details about filtering rules, nat rules and routing.

The target twin stores a configuration table containing details about all the modules and their instances. Each module corresponds to a row in this table, storing two sets of values: one for configuration parameters and another for vulnerabilities. Instances with the same module configuration share the same vulnerabilities. Additionally, the target twin includes a mapping table that associates each configuration table row with the infrastructure nodes running the described instances.

For each vulnerability of an infrastructure component, the target twin describes each attack it enables and pairs each attack with a set of attributes that includes both pre- and post-conditions and the success probability. The attack pre-condition is the set of access rights an attacker needs to implement it. Instead, the post-condition is the set of access rights the attacker acquires if the attack is successful. Pre- and post-conditions determine how a threat actor can chain attacks in its intrusions.

The Haruspex predictive platform collects the information to build the target twin from a system inventory or by running a vulnerability scanning of all the system nodes. The physical topology of the infrastructure and the logical one are built from routing tables, and firewall rules.

The other twin of interest in relation with H-Verify is the one of an attacker or threat twin. This twin models an intelligent and adaptive attacker that has an explicit goal, a set of access rights it aims to acquire while minimizing its efforts. The threat twin describes the goal, the attack surface, and the initial access rights of the attacker. Further information describes the strategy the attacker applies to select an action. The attacker strategy (Hutchins et al.,

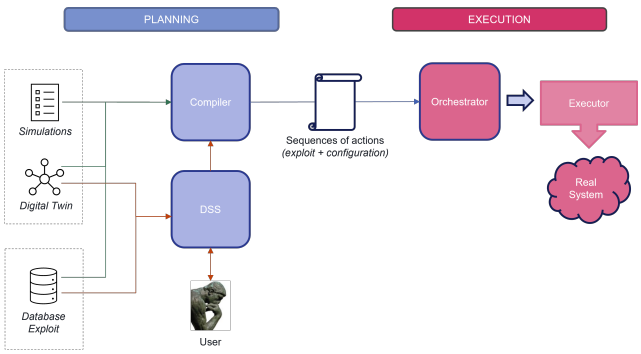

Figure 1. H-Verify Architecture

2011; Maigre, 2022) chooses the next action to be executed as a function of the goal, the access rights the attacker currently owns, and the information on the infrastructure it has acquired by its previous actions. The strategy also determines how the attacker handles the failure of an action.

3.2. Adversary Simulations

The abstraction level of the twins enables the platform to implement adversary simulation and model the action an attacker selects and executes in any situation according to its strategy. For each action, the platform records the target component, the node running it, and, if the action is an attack, the enabling vulnerability. To return the overall time to implement an intrusion, the platform also records the time to execute an action.

The output of each simulation is stochastic as it depends on the success or the failure of the attacks. An attack may fail either because of its complexity and/or for factors the attacker cannot control. The predictive platform determines the favorable outcome of an attack according to its success probability. An attack may also fail because of a detection mechanism that protects the target component (Khraisat et al., 2019). The detection of an attack does not imply the failure of the intrusion if the attacker has previously gained some persistence in the target infrastructure.

Each simulation returns the actions the attacker has executed and if it has reached the final goal. Due to a poor strategy, an attacker may also execute some actions that are useless to reach the goal. For this reason the simulation also returns the plain sequence of all and only the useful actions to reach the goal: we refer to this sequence as an intrusion.

4. Architecture

In the following we describe the architecture of H-Verify and how the user can interact with the platform. The software architecture of H-Verify is shown in Figure 1.

H-Verify consists of three core modules:

- Compiler, used in the planning phase.
- Decision Support System (DSS), used in the planning phase.
- Orchestrator, used in the execution phase.

The planning phase consists of any operation of H-Verify that does not directly involve the target system but prepare the sequence of actions against the target. Possible operations in this phase are design, compilation, modification of sequence of actions, and model management.

The execution phase includes all the actions that H-Verify executes and that affect directly the target system. This phase executes the planned and configured actions, including attacks, with an actual impact on the target system.

4.1. Compiler

The Compiler maps the adversary simulations that the predictive platform has run using the digital twins into real sequences of actions against the target system. To this purpose, it requires two inputs:

- The digital twin of the target system and the adversary simulations by the predictive platform.
- An exploit database that provides the exploits to take advantage of some of the vulnerabilities the adversary simulations have exploited.

For every simulation step performed by the predictive platform, H-Verify:

1. checks the exploit database to identify those that can implement the attacks in the simulation.
2. identifies the information on the attack to configure the corresponding exploit.

Different actions can use different properties from the digital twin. Most common ones can be:

- IP address of the target.
- Port and protocol, which are also used to further refine the exploit selection. Some vulnerabilities (e.g., shellshock) can affect different class of services (ftp, web servers, etc.). H-Verify can use the digital twin data to specialize the attack on the vulnerable service.
- Payload choice and configuration. Topology data is used to configure a reverse or bind shell, according to the real system filtering rules.

Other information as target architecture, vulnerable URL, credentials and more, are retrieved from the digital twin and used to configure the attacks.

If multiple exploits are available for a single vulnerability, the tool compiles all of them to offer to the user multiple choices. All these steps are properly linked to configure a completely automated sequence of actions.

The result of the Compiler is the list of all the sequences that it can produce starting from the adversary simulations

using the digital twin and the exploit database. These sequences are stored in a database for future execution or further user customization.

4.2. Decision Support System

Whereas the Compiler focuses on automating the execution of action sequences of the adversary simulations, the DSS works as a user-driven intelligence. Users interact with this module to design or customize their own sequences, by querying the alternative steps the tool can implement against the real system, and have the sequences prepared for the execution.

An advanced query system allows the user to search and identify information of interest. Since the digital twin stores all the information of interest on the infrastructure such as vulnerabilities, topology, subnets, services, the user can query the DSS to identify the desired attack surface. To achieve this, different filters are available for the user:

- Topology-based: the user can view the networking communication of the system and use it to ask the DSS various information, for instance which devices are exposed on the internet or those that can be reached starting from a specific infrastructure node.
- Structural-based: the user can view information regarding software and hardware of the infrastructure. Many queries can be implemented by combining this information, for example which hypervisor are susceptible to attacks or the web server running on windows architecture.
- Vulnerability-based: the user can view vulnerability information about the system and use it to identify distinct threats, for instance which devices are susceptible to remote attacks or those that can be attacked by a worm (e.g., WannaCry)

For every user query, the DSS returns the actions that it can build using the current exploit database. The user can select the actions of interest and the module automatically prepares them for future executions, as in the case of the Compiler module.

The DSS can work in two different modes:

- Step-by-step: the user decision process is supported step by step. The user can initially define the first action, and then the DSS intelligence suggests the next ones.
- Multiple steps: the user can specify a target, and the DSS will suggest the list of alternative starting points from which it can build a sequence to compromise the target. As seen for the Compiler module, the whole sequence can be automatically prepared. If the user specifies a starting node in the infrastructure, the DSS can suggest all the possible targets that can be reached from the node.

The DSS module can create new sequences of actions or customize existing ones, even those created by the Compiler modules. A local database stores the output sequences for future execution or further customization.

4.3. Orchestrator

The Orchestrator executes sequences of steps that include both attacks and actions. These sequences may be produced by the Compiler or customized by the user through the support of the DSS.

The input for the Orchestrator consists of an ordered sequence of steps, where each step represents a logical action and contains one or more modules that implement this action. For instance, a step might represent the exploitation of a particular system vulnerability, for which multiple attacks (exploits) are available.

Logically, steps can be classified into three categories:

- Vulnerability exploitations: if executed successfully, they either open a session or execute a command on the target machine.
- Post-exploitation actions: execute modules for network discovery, persistence, exfiltration, resource upload, pivoting, port forwarding, etc.
- Session command: executes a command on the console of a session opened by a previous step.

When alternative modules exist for the same step, the Orchestrator will execute them one by one in the order they were configured until one is successful. At this point, the Orchestrator assumes the step is successful, and it passes to the next step, without executing the remaining modules of the same step. If a module fails, the Orchestrator will attempt to repeat it a given number of times - configurable by the user - before considering it failed and moving on to the next one for the step. If all modules of a step fail, the sequence fails.

Before executing a sequence, the user can configure the following parameters, where all the times are in seconds:

- MaxActionWaiting: the maximum time for the execution of a module. When this time expires, the module execution is interrupted.
- MaxActionAttempts: the maximum attempts (in case of failures) for a single module before moving to the next one (if present).
- MaxBacktrackings: the maximum attempts for backtrackings before the failure of the sequence. A backtracking occurs when a previously opened session is unexpectedly closed, and it is necessary to start again from the step that opened it.
- SleepBetweenActions: time between the execution of a module and the next one, whether it is a repetition of the same module or the first attempt to execute the next one.
- SleepBeforeNewSession: time before verifying that a new session has been successfully opened and it is stable after an exploit execution.

These parameters can implement alternative adversary

strategies that produce distinct noise levels. This can be helpful to test the intrusion detection systems of an infrastructure against advanced threats.

Before starting a sequence, the user can choose the operational mode of the Orchestrator:

- Automatic, where it autonomously performs all the actions of the selected sequence.
- Guided, where it waits for user instructions before executing any action.

It is possible to switch between these modes at any time, even during execution. The Orchestrator can autonomously verify whether open sessions are stable, as well as detect any connection interruptions to re-establish them by repeating any required step through a backtracking mechanism.

The user can interact with open sessions at any time, to execute console commands, exfiltration operations, and/or to upload resources to the compromised host. Furthermore, the user can consult the DSS to add and configure new attack steps in the guided manner described in Subsection 4.2.

Every event generated during execution is logged with its time of occurrence. For each executed module, the configuration, output text, success or failure, and whether and which type of session was opened (and on which host) are recorded. All user interactions with the sessions (command text and obtained output) and with the Orchestrator itself (e.g., pauses and resumes of the execution, change of operational mode, etc.) are also recorded. Events have a hierarchical structure: for example, the execution of an exploit will be generated from another event, namely the execution of the step it belongs to. These logs are accessible both during execution, as they are generated in real-time, and afterward, when the sequence ends, in the form of an attack report enriched with other significant statistics.

Moreover, the information in the recorded logs is used during (and after) execution to enrich the digital twin of the attacked infrastructure. This supports the enhancement of the DSS and the execution of more accurate simulations, improving the suggestions to the user during the engagement or in future attacks.

The Executor, shown in Figure 1, is a third-party software that actually implements exploits.

At the end of a sequence execution, the user may clean up the state of the attacked machine and decide whether to repeat the full sequence or to execute new ones.

5. Results

In this section, we present some results of two distinct test infrastructures:

1. A virtual laboratory: a small yet deep network to test the capabilities of H-Verify, mainly the automatic creation and execution of attack sequences.

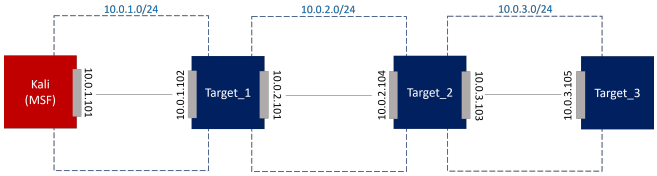

Figure 2. Virtual Laboratory

2. A real-world infrastructure: to test scalability on medium-sized systems, assess the actual potential of the DSS, and discover any weakness of H-Verify.

These experiments have used the Metasploit framework (Rapid7, a) as the Executor module and exploit database. Communications between the Orchestrator and Metasploit were implemented through the latter's RPC server.

5.1. Virtual Laboratory

As the main test scenario, we set up a virtual infrastructure that is easily replicable but offers all the key scenarios we are interested in. The most interesting features are:

- Network depth, to test whether H-Verify can autonomously determine which actions, such as routing or port forwarding, are required to penetrate defense in depth systems, protected by multiple firewalls. This also allowed us to evaluate the backtracking mechanism H-Verify offers and the pivoting capabilities of the Metasploit sessions.
- Highly vulnerable machines to test whether the H-Verify Compiler can automatically build complete sequences of actions starting from the output of the adversary simulations on the digital twin of the virtual infrastructure.

Figure 2 shows the virtual laboratory. Some details:

- The Executor module, Metasploit in this experiment, runs on the red-colored Kali machine. In both simulations and real attacks, it represents the attacker's starting point.
- The attacker's machine can only communicate with Target_1.
- Target_1, Target_2, and Target_3, i.e., the blue-colored hosts, are Metasploitable2 machines (Rapid7, b), highly vulnerable by design.
- Target_1 and Target_2 have a dual interface, each connected to a distinct subnet. They act as firewalls and forbid communication between machines.

In this scenario, to compromise Target_3, we should:

1. successfully attack Target_1,
2. perform a forwarding action to Subnet_B,
3. compromise Target_2,
4. perform another forwarding action to Subnet_C,
5. successfully attack Target_3.

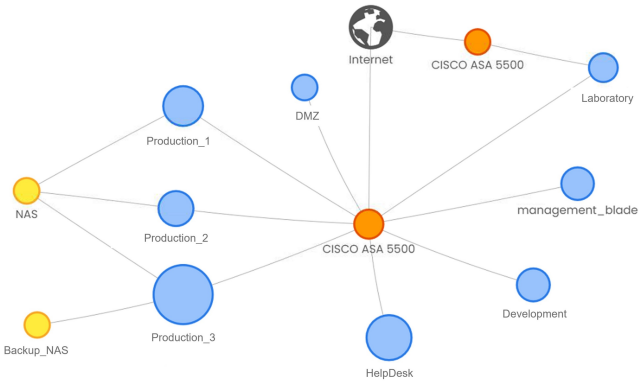

Figure 3. Real Infrastructure Topology

During these steps, we have to maintain stable communications between the opened sessions.

The creation of the digital twin required just a few Nessus scans of the network. Then we ran 200.000 attack simulations that led to 21.786 alternative intrusions which exploited 275 distinct vulnerabilities.

H-Verify has compiled 42 sequences of actions that use 30 distinct exploits.

To better understand the gap between the output of simulations and the distinct sequences that H-Verify can compile, it is important to highlight that simulations can include even non-automatable actions such as man-in-the-middle or cryptographic attacks. These actions cannot be filled in automatically by H-Verify, but require user interaction, which can occur at a later stage. In the compiling phase the focus is on the complete automation of the sequence.

Obviously, by increasing the number of simulations on digital twin we produce a larger number of intrusions and therefore more automatable H-Verify sequences of actions. Still, in a few minutes, the user can produce tens of ready-to-use sequences of actions where the only required know-how is the ability to run some Nessus scans.

5.2. Real Infrastructure

The second test scenario has considered a real-world infrastructure, consisting of 316 hosts, 8 subnets, and 2 CISCO ASA firewalls. Figure 3 shows this network.

Another interesting feature of this system is its heterogeneity, both in terms of operating systems (Linux, Windows, MacOS) and of hosts (servers, firewalls, hypervisors, IDS, etc.), as well as in the applications and services it runs.

The main goal of this test was to evaluate the potential of the DSS in:

- Supporting the user in configuring single-step but also multi-step actions in complex situations with wide maneuver possibilities.
- Properly leveraging information on network topology, accurately determining intrusions that exploit logical connections resulting from the routing and filtering rules in the real system.
- Scaling to medium and large infrastructures.

The creation of the digital twin has required some Nessus scans of the whole system and the import of the firewall configurations. Then, we ran 180.000 simulations that discovered 4.405 alternative intrusions that exploited 194 distinct vulnerabilities. H-Verify has compiled 65 attack sequences using 33 distinct exploits.

Since this is a real system, regularly updated and patched, only a few exploits in the Metasploit database were available to use. This has reduced the number of complete sequences of actions that H-Verify has been able to compile in an automated way from the digital twin simulations. In Section 6 we analyze this issue in more details.

However, the DSS has been able to offer an excellent support, meeting expectations, and managing to bridge much of the gap due to lack of exploits. In fact, DSS has pointed out 37 distinct targets for fully automated multi-step sequences of actions. Furthermore, the DSS was able to detect 291 (of 316 hosts) as possible starting points for automated sequences of actions.

These tests have confirmed the advantages that H-Verify offers when targeting heterogenuous systems with distinct sizes. The information in the digital twin and in the output of the simulations, combined with the automation of the Compiler, drastically reduces the abilities required to undertake offensive actions. In some contexts, H-Verify can also be an excellent training tool. The DSS may be a great ally for human operators managing complex systems, as it strongly reduces the time and complexity of operations.

6. Limitations

Some issues affecting the effectiveness and applicability of H-Verify are not unique to our tool but are rather prevalent across the broader spectrum of offensive security applications.

The main problem is the size of the exploit database that H-Verify can access as it strongly influences the number of fully compiled sequences of actions. The number and quality of exploits and procedures are critical to implementing meaningful and realistic red teaming campaigns. However, there exists a widening gap between the discovery of vulnerabilities and the publication of exploits. As illustrated in Figure 4, over the past three years (2021-2023), more than 20.000 Common Vulnerabilities and Exposures (CVEs) have been published annually (SecurityScorecard), while the total number of exploits published from January 2021 to mid-2023 on primary sources like Metasploit, GitHub (GitHub), and ExploitDB (OffSec) is approximately 6.000 as shown in Figure 5 and Figure 6 (Jacobs). This tenfold discrepancy may be attributed to the degree of automation and ethical considerations. While the discovery of vulnerabilities can be extensively automated and is encouraged within the community for enhancing risk aware-

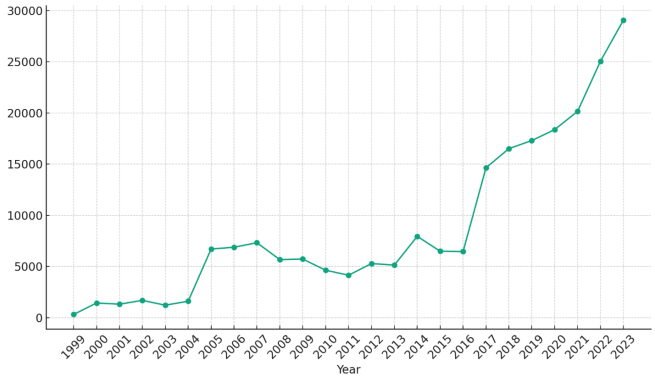

Figure 4. CVEs published by year

Source is from this website: https://www.cvedetails.com/browse-by-date.php

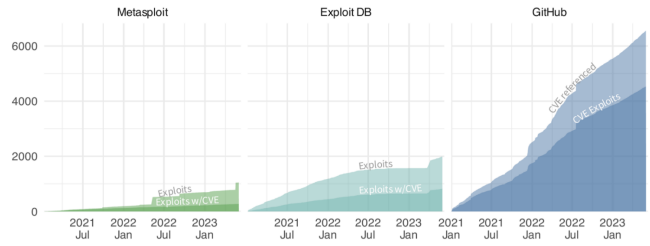

Figure 5. Overlap Among Sources of Published Exploits from January 2021 to mid-2023

Source is from this website: https://www.cyentia.com/exploring-exploit-intelligence-service/

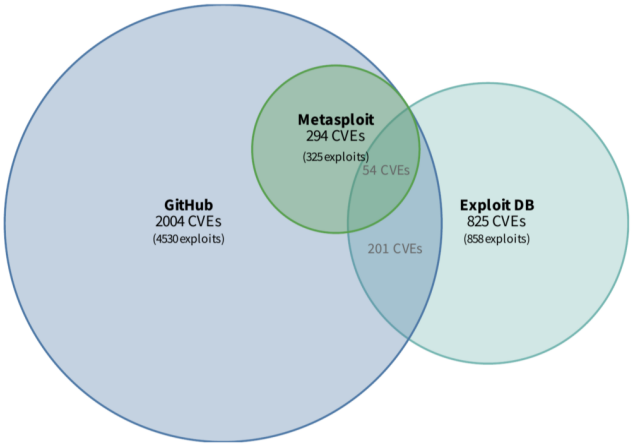

Figure 6. Published Exploit Code by Source from January 2021 to mid-2023

Source is from this website: https://www.cyentia.com/exploring-exploit-intelligence-service/

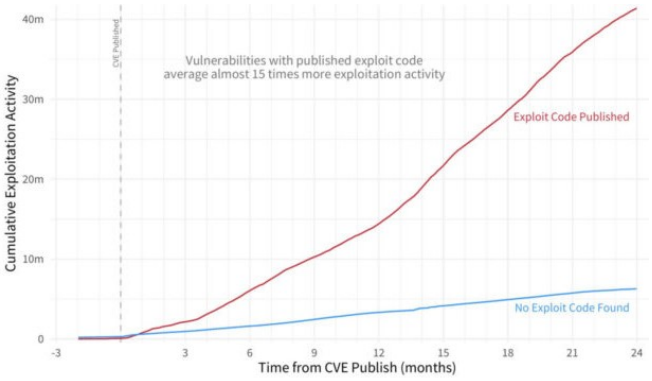

Figure 7. Exploitation Activity Between Vulnerabilities With and Without Exploit Publication

Source is from this website: https://www.cyentia.com/enhancing-threat-driven-remediation/

ness, the development of new exploits remains largely a manual, time-consuming, and therefore expensive task. Furthermore, the potential profit from the malicious use or the sale of the exploits themselves strongly hinders their public release. The ethical issue regarding their publication also acts as a deterrent. Indeed, Figure 7 confirms that vulnerability exploitation significantly increases after a proper exploit is published (Gimarelli).

In our tests, the Executor module was the open-source version 6.3.55 of Metasploit, which integrates 5.511 procedures, including 2.397 exploits. However, the sheer availability of exploits is not the sole challenge; their integration, testability, and reliability also play pivotal roles in the practical use of offensive security tools. While extensive public collections like ExploitDB exist, with over 45.000 entries, not all are validated or tested, and many require manual integration, demanding significant expertise and time. Among commercial solutions, Core Impact (Fortra, b) arguably possesses the most extensive framework of tested, validated, and integrated exploits, numbering around 3.350.

The reliance on well-known open-source projects like Metasploit is advantageous in terms of development and required knowledge but, as a drawback, current Antivirus (AV) and Intrusion Detection Systems (IDS) can easily identify and block these tools. This greatly reduces their effectiveness in well-organized real-world environments.

In light of these limitations, the operator's experience can still be an added value in the practical use of H-Verify. Expertise may be required to complete partial compilations of simulated plans or to implement advanced encoding and evasion strategies, leveraging the open-source tools underpinning our solution in well-defended environments.

7. Comparison with other offensive security tools

This section compares H-Verify against the current main solutions in the offensive security space. The landscape includes tools such as Metasploit, Metasploit Pro, and Core Impact, which are prominent in penetration testing, and Caldera (MITRE, b), Sliver (BishopFox) Cobalt Strike (Fortra, a), Metta (Uber), and APT Simulator (NextronSystems), which are notable in adversary simulation/emulation. Additionally, Infection Monkey (Aka-

maiTechnologies) is recognized in the context of breach and simulation. We also mention tools with less functionalities, such as Pupy RAT (n1nj4sec) and Powershell Empire (EmpireProject), in the domain of post-exploitation, alongside several red team test collections, including Atomic Red Team (RedCanary), Stratus Red Team (Datadog), Dumpster Fire (TryCatchHCF), and Red Team Automation (EndGame). While these tools sometimes evade precise categorization and possess overlapping features, we accept the self-categorization each tool provides.

To our knowledge, H-Verify currently stands unique in fully leveraging the potential of digital twin technology and adversary simulations. This capability allows H-Verify to automatically build optimal sequences of actions and provide a robust and advanced DSS during the planning phase, improving and extending the partial adoption of the digital twin technology in tools such as Metasploit Pro and Core Impact. These tools only create partial digital twins of the target infrastructure, using information gathered during attacks, thus yielding incomplete vulnerability awareness with respect to the overall system analysis that H-Verify offers.

Furthermore, the ability of running adversary simulations where attackers adopt alternative strategies and have distinct priorities enables the implementation of multiple attack strategies and malware-spreading scenarios, a feature paralleled only by Infection Monkey in terms of malware-spreading simulation. The ability of H-Verify to autonomously execute sequences of actions and to interleave forwarding and routing operations between sessions as necessary, results in a level of automation that only Caldera, Metta, and APT Simulator partially match. To a lower extent, this is matched by Metasploit Pro and Core Impact.

These features arguably make H-Verify the best tool to support even red teams without a deep expertise—a distinction potentially matched only by Core Impact. During the test phase, the Executor module of H-Verify was the open-source version of Metasploit, inheriting its collection of procedures (including exploits) alongside evasion, encoding, and Command and Control (C2) capabilities. Here, Core Impact generally outperforms in the realms of exploit quantity and quality, while Sliver and Cobalt Strike excel as C2 frameworks, with Metasploit Pro offering additional functionalities in encoding and evasion.

Altogether, while maintaining the categorization inherent to each tool, H-Verify results in significant advancements in offensive security practices. Its unique integration of full digital twin technology, comprehensive decision support system, and automated execution of sequence of actions including attacks sets a new standard in the field, positioning it as a potentially transformative tool for practitioners of varying expertise levels.

8. Future Work

To enhance H-Verify's capabilities, extend its applicability, and improve its effectiveness in real-world scenarios, several avenues exist:

1. Enhancing encoding and obfuscation capabilities: Integrating encoding and obfuscation tools into the H-Verify ecosystem, along with new technologies based on reinforcement learning, to augment the likelihood of successful red team engagements in well-organized, real-world systems.
2. Integration of External Exploits: Recognizing that some attacks require exploits not in the current database, we aim to simplify import from other external sources such as GitHub and ExploitDB. This will support users in planning by providing access to a broader range of attack vectors and solutions, thereby enhancing the versatility and applicability of the tool to diverse security challenges.
3. Attack Infrastructure Deployment: Following the successful compromise of a system, we plan to explore the deployment of a controlled botnet, enabling the installation of interconnected modules for persistence and the coordination of synchronized remote commands. This would potentially enhance the C2 capabilities of the tool through the integration with frameworks like Sliver. Such features extend the potentials of H-Verify beyond initial penetration, offering a comprehensive toolkit for long term persistence in compromised systems.
4. Incorporation of APT Profiles: To bolster the product's adversary emulation capabilities, we plan to add real Advanced Persistent Threat (APT) profiles to the implementation of compiled attacks. This enables users to simulate and defend against highly sophisticated and targeted attack scenarios, enhancing the realism and relevance of security assessments using H-Verify.

9. Conclusion

This paper has presented H-Verify, a new offensive security tool, set against the backdrop of established instruments such as Metasploit, Core Impact, Cobalt Strike, Caldera, and Sliver. The adoption of digital twin and adversary simulations technologies is the most innovative feature of H-Verify. This extends its role beyond traditional offensive applications to embrace roles in purple team engagements and infrastructural risk evaluation.

As a "Verify" tool, H-Verify is designed to show the feasibility of using the output of simulations on digital twins to discover real intrusions, thereby providing a novel approach to assessing the robustness of infrastructure, whether in-house or that of suppliers. This results in a more informed validation of the infrastructural integrity and the effectiveness of the selected countermeasures. The tool potential extends to critical security aspects, making it an invaluable resource for organizations seeking to fortify their defenses against increasingly sophisticated cyber threats.

Using H-Verify a purple team can test the infrastructure security offering a hands-on approach to personnel training and the evaluation of the effectiveness of defensive tools. This dual functionality is an important contribution to the development of comprehensive security strategies, ensuring that both offensive and defensive measures are examined and optimized.

10. Acknowledgements

The authors extend their sincere gratitude to Ing. Marcello Montecucco, CEO, and Amm. Dino Nascetti, co-founder, of Haruspex, for their generous support and insightful feedback, which have greatly contributed to the refinement of this research.

References

AkamaiTechnologies. Infection monkey. https://www.akamai.com/it/infectionmonkey [Accessed: April 10, 2024].

Baiardi, F. (2019). Avoiding the weaknesses of a penetration test. *Computer Fraud Security*, 2019:11–15.

Baiardi, F. (2023). Merging fmea and digital twins to improve trustfulness. pages 1–7.

Baiardi, F. and Sgandurra, D. (2013). Assessing ict risk through a monte carlo method. *Environment Systems and Decisions*, 33.

Baiardi, F. and Tonelli, F. (2021). Twin based continuous patching to minimize cyber risk. *European Journal for Security Research*, 6:1–17.

Baiardi, F., Tonelli, F., and Bertolini, A. (2015). Cyvar: Extending var-at-risk to ict. pages 49–62.

BishopFox. Sliver. https://bishopfox.com/tools/sliver [Accessed: April 10, 2024].

Datadog. Stratus red team. https://stratus-red-team.cloud/ [Accessed: April 10, 2024].

EmpireProject. Empire. https://github.com/EmpireProject/Empire [Accessed: April 10, 2024].

EndGame. Red team automation (rta). https://github.com/endgameinc/RTA [Accessed: April 10, 2024].

Fortra. Cobalt strike: Software for adversary simulations and red team operations. https://www.cobaltstrike.com/ [Accessed: April 10, 2024].

Fortra. Core impact: Penetration testing software to safely uncover and exploit security weaknesses. https://www.coresecurity.com/products/core-impact [Accessed: April 10, 2024].

Gimarelli, C. Enhancing threat-driven remediation: Prioritizing vulnerabilities with exploit intelligence. https://www.cyentia.com/enhancing-threat-driven-remediation/ [Accessed: April 10, 2024].

GitHub. Github. https://github.com/ [Accessed: April 10, 2024].

Husák, M., Komárková, J., Bou-Harb, E., and Čeleda, P. (2019). Survey of attack projection, prediction, and forecasting in cyber security. *IEEE Communications Surveys Tutorials*, 21(1):640–660.

Hutchins, E., Cloppert, M., and Amin, R. (2011). Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Leading Issues in Information Warfare Security Research*, 1.

Jacobs, J. Exploring exploit intelligence service (eis): Tracking exploit code. https://www.cyentia.com/exploring-exploit-intelligence-service/ [Accessed: April 10, 2024].

Kavak, H., Padilla, J. J., Vernon-Bido, D., Diallo, S. Y., Gore, R., and Shetty, S. (2021). Simulation for cybersecurity: state of the art and future directions. *Journal of Cybersecurity*, 7(1):tyab005.

Khraisat, A., Gondal, I., Vamplew, P., and Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 2.

Langlotz, P., Klar, M., yi, l., Hussong, M., Sousa, F., and Aurich, J. (2022). Concept of hybrid modeled digital twins and its application for an energy management of manufacturing systems. *Procedia CIRP*, 112:549–554.

Lehner, D., Pfeiffer, J., Tinsel, E.-F., Strljic, M., Sint, S., Vierhauser, M., Wortmann, A., and Wimmer, M. (2021). Digital twin platforms: Requirements, capabilities, and future prospects. *IEEE Software*, PP.

Maigre, M. (2022). Cyber threat actors: how to build resilience to counter them. *Hybrid CoE*, (11).

MITRE. Att&ck matrix for enterprise. https://attack.mitre.org/ [Accessed: April 10, 2024].

MITRE. Caldera: A scalable, automated adversary emulation platform. https://caldera.mitre.org/ [Accessed: April 10, 2024].

n1nj4sec. Pupy. https://github.com/n1nj4sec/pupy [Accessed: April 10, 2024].

NextronSystems. Apt simulator. https://github.com/NextronSystems/APTSimulator [Accessed: April 10, 2024].

OffSec. Exploit database. https://www.exploit-db.com/ [Accessed: April 10, 2024].

Rapid7. Metasploit. https://www.metasploit.com/ [Accessed: April 10, 2024].

Rapid7. Metasploitable 2. https://docs.rapid7.com/metasploit/metasploitable-2/ [Accessed: April 10, 2024].

RedCanary. Atomic red team. https://redcanary.com/atomic-red-team/ [Accessed: April 10, 2024].

SecurityScorecard. Browse vulnerabilities by date. https://www.cvedetails.com/browse-by-date.php [Accessed: April 10, 2024].

Tao, F., Zhang, H., Liu, A., and Nee, A. Y. C. (2019). Digital twin in industry: State-of-the-art. *IEEE Transactions on Industrial Informatics*, 15(4):2405–2415.

TryCatchHCF. Dumpsterfire toolset. https://github.com/TryCatchHCF/DumpsterFire [Accessed: April 10, 2024].

Uber. Metta. https://github.com/uber-common/metta [Accessed: April 10, 2024].