# Cyber range automation, a bedrock for AI applications

Leonardo Gavaudan[1], Swann Legras[1] and Véronique Ventos[1]

[1]NukkAI, 75013 Paris

**Abstract**

This paper proposes an automated solution for conducting cybersecurity research. It shows how automation can improve the foundations of cybersecurity research, and consequently facilitate and bolster the development of artificial intelligence applications. The challenges that cybersecurity researchers face are discussed, as well as what automation offers to tackle them at three different stages: provisioning, configuration, and attack simulation.

**Keywords**

cyber range, security automation, security testing, adversary emulation, threat hunting, infrastructure-as-code

## 1. Introduction

Over the last few years, the pace of cyber attacks has particularly accelerated; their complexity and reach have relentlessly been growing. The 2020 Solarwinds attack, an attack estimated to have infiltrated thousands of organizations among which United States government systems, is perhaps the best example of the trend. From the start of 2020, the following major attacks can be already named: Solarwinds, Colonial Pipeline, JBS, Microsoft Exchange Servers. The growing recognition of the need for artificial intelligence applications in the realm of cybersecurity research in order to respond to the increased complexity and frequency of these attacks, is paralleled with a lack of a good ecosystem for them to flourish. The current cybersecurity research process has more of a manual approach, and is not best suited for artificial intelligence development.

According to the European Defense Agency, a cyber range is "a multipurpose environment in support of 3 primary processes: knowledge development, assurance and dissemination" composed of "three complementary functionality packages": a Cyber Research Range (CRR), a Cyber Simulation & Test Range (CSTR), and a Cyber Training & Exercise Range (CTER)[1]. It is a common issue for cybersecurity researchers who want to study a particular attack or technique to end up realizing just how incredibly arduous the process of setting up such a cyber range is. Cybersecurity professionals looking to get started with AI development, and AI researchers looking to develop applications are confronted with a same problem. Without access to both a repository of good datasets and a system to keep it up to date, the mission for developing production ready, up to date and advanced AI applications remains an extremely difficult task. An automated cyber range is a cyber range where the deployment and configuration of the infrastructure, and initial installation of red teaming tools are automated. By automating its setup, cybersecurity and AI researchers are empowered to focus on studying the core of an attack, and building AI applications, rather having to worry about the underlying groundwork.

Given that the bottle neck to AI research and development is a lack of good datasets, the aim of this paper is to illustrate an automated cyber range platform that researchers can use to easily generate and access high quality, and diverse attack simulation datasets. The contribution of this paper is in the showcase of how and why existing open source technologies can be assembled to build an end to end platform solution for cybersecurity automation. This paper explains how the choice for each component of the solution is justified. More importantly, it will compare the platform solution as a whole to the current state of practice for end to end automation solution. As much as this paper is an abstract and theoretical explanation for cybersecurity automation, it is also a practical guide. That is why the paper will instantiate the proposed solution through an advanced persistent threat simulation example. The advanced persistent threat example will help depict the technologies, as well help researchers to start implementing, and using them.

The plan of the paper is as follows: To begin with, in section 2, we take a look at the APT29 attack simulation example, and current end to end solutions. In section 3, we inspect the current way cybersecurity research takes place and its shortcomings. We then provide, in section 4, a comprehensive automated solution that addresses the challenges discussed in the previous section. Finally, in section 5, we examine how cybersecurity automation positively impacts artificial intelligence development.

*CESAR 2021: Automatisation en Cybersécurité - Automation in Cybersecurity*

EMAIL: lgavaudan@nukk.ai (L. Gavaudan); slegras@nukk.ai (S. Legras); vventos@nukk.ai (V. Ventos)

## 2. Prerequisites & related works

In this section, we look at the context from which the scenario of APT29 is drawn, and why it is fit to describe the cybersecurity automation landscape in 2.1. We then look at two existing end to end solutions, one of whom represents current state of the art and practice in 2.2. Both the pros and cons of the existing end to end solutions are analyzed, as well as how the solutions compare to the one proposed in this paper.

### 2.1. APT 29

MITRE Engenuity is a technology foundation and a portal through which MITRE collaborates with the private sector, and applies state of the art innovation that emerges out of their research and development activity[2]. Operating in MITRE Engenuity is the Center for Threat Informed Defense (CTID), a "privately funded research and development organization"[3] that has developed an adversary emulation library of advanced persistent threat attack plans. If an autonomous red teaming tool allows us to launch adversary simulations, then the plans produced by CTID are the important inputs needed to generate high quality attack simulation datasets.

One of the attack simulation plans that CTID has developed is a plan to simulate an APT29 attack. APT29 is a "threat group that has been attributed to Russia's Foreign Intelligence Service (SVR). They have operated since at least 2008, often targeting government networks in Europe and NATO member countries, research institutes, and think tanks"[4]. By extensively studying real cyber attacks conducted by APT29, the CTID team built an attack plan that simulates and draws on identical or similar techniques, tactics and procedures that the Russian hacking group has previously used. We have chosen this particular attack simulation plan because the attack's complexity will further accentuate the benefits of automation. The APT29 simulation plans provide both a manual and automatic implementation guide for how to carry out the attack simulation. This in turn allows us to start setting quantitative benchmarks for how much gain in time and productivity one can achieve through an automated solution.

The CTID team continues to develop new emulation plans, and refine already established ones. The plans are mapped to MITRE's ATT&CK framework, and used to evaluate detection solutions both in the context of the 'ATT&CK Evaluations', a detection solution evaluation series led by MITRE, and for individually and separately evaluating one's own detection solution. This ecosystem allows researchers to develop applications on the emulation library with confidence.

### 2.2. Existing Cyber Ranges

#### 2.2.1. Splunk - Attack Range

The first end to end solution proposed is Splunk's Attack Range[5]. Introduced in 2019 and developed by the Splunk Threat Research Team, Attack Range is the state of art and state of practice when it comes to integrating multiple technologies to automate each stage in the cybersecurity research process. It takes care of deploying and configuring infrastructure in the cloud, to running autonomous adversary emulation and extracting the resulting logs. The Attack Range technology stack is very similar to the one proposed in this paper, albeit the solution presented here proposes some improvements. Attack Range uses Terraform to deploy infrastructure in either AWS or Azure's cloud, and Ansible to then configure it. Attack Range also deploys MITRE's CALDERA as an autonomous red teaming tool. Lastly, it uses Window's WEF technology in order to recuperate the logs generated from an attack. Attack Range then indexes the logs on a Splunk Server and uses other Splunk technologies for security orchestration and rule based detection.

However, although Splunk's Attack Range allows us to test individual abilities and techniques, it does not provide comprehensive attacks for researchers to study, run, and develop on. Likewise, Attack Range's default environment setup is composed of 2 Windows machines, and 2 Linux servers. In order to execute complex attack chains that require larger and convoluted environments, one would need to firstly adapt to required the environment by editing a configuration file provided in Attack Range. Then, one would need to gather, in the correct order from the MITRE ATT&CK framework, the list of technique IDs used in the attack, and feed it as a command line argument argument when initiating the program. Depending on an attack's complexity, the configuration file can quickly become bloated, hard to manage, and work against the initial intended goal that the research team had set: facilitating cybersecurity automation for threat research.

Secondly, because Attack Range was developed by a research team within Splunk, the data indexing and visualization step, and SOAR step of the solution can only be configured with Splunk technologies: Splunk and Splunk Phantom. Although Splunk is recognised as a leader in the field of log collection, analysis, and detection, avoiding vendor lock-in is a fundamental concept to consider when it comes to designing an end to end solution. It allows the solution to be easily modified in order to best fit the needs of users, and ensures that the solution doesn't have a single point of failure.

Lastly, Splunk's red teaming automation is based on CALDERA's old version 2 release, and therefore any new features developed and introduced in CALDERA will not be compatible with Attack Range.

#### 2.2.2. Microsoft - SimuLand

The second tool we're going to a look at is SimuLand[6], a tool developed by Microsoft and introduced in 2021. The tool isn't a fully automated end to end solution. It automates the deployment and configuration of infrastructure, but lacks red team automation tooling. Rather, it aims to guide researchers on how to manually simulate different techniques on already deployed and configured infrastructure. It deploys and configures an infrastructure in the cloud using Azure Resource Manager Templates, and has good integration with different security, DevOps, and cloud products within the Microsoft ecosystem (Microsoft 365 Defender, Azure Defender, and Azure Sentinel). On another hand, SimuLand's design limits users to using Azure as a sole cloud provider, which is particularly problematic when studying security exploits directly embedded in a cloud provider's system.

## 3. Manual security research

In this section we will look at how current cybersecurity research is being conducted at three different stages: provisioning (3.1), configuration (3.2), and attack simulation (3.3). The section will analyze in detail the procedure that a researcher would go through for each step, and analyze its weaknesses and disadvantages, all done through the lens of a setup for an APT29 attack simulation.

### 3.1. Provisioning

The first step in conducting cybersecurity research is the deployment of an environment, a cyber range, in which we want to test out different abilities and techniques. A common way to manually deploy the infrastructure is by requesting the necessary resources through a cloud provider website's graphical user interface. Another solution is to build the required infrastructure with virtualizing software like VirtualBox, or KVM and locally host the environment.

Going through a cloud provider entails having to separately request each resource and specify instance details. The process of setting up an environment for a complex attack can call for requesting to the cloud provider virtual machines, networks, sub-networks, network interface controllers, network peerings, and additional resources. For each of these resource requests, one must specify a group of settings. For instance, a virtual machine will typically require the disk, image, CPU, RAM among other details to be provided in order to be deployed. Individually deploying resources is prone to error, hard to debug, tedious, and time consuming.

In order to set up the correct environment for the APT29 scenario as depicted in Figure 1 (1), a target and attacker environment is required. The target environment is a Windows domain sub-network with 2 Windows servers with the '2019-Datacenter' SKU, one serving as a domain controller, and the other serving as a file server, and three Windows workstations running Windows 10 1903 with a "19h1-pro" or "1903-evd-o365pp" SKU. The domain controller and file server are usually controlled and under the supervision of IT professionals, whilst workstations usually represent regular computers that non-technical employees use. In this case, all VMs are "Standard B4MS" instances, with four vCPUs and 16GB of RAM. The attacker environment is a second sub-network with 2 Linux machines running 18.04.3 LTS Ubuntu, one serving as a traffic redirector, and the other as a C2 (Command and Control) from where attack commands are sent. The sub-network also has a workstation running Windows 10 1903 with the same SKU as the target workstations, it serves as a platform to replicate the target environment, appropriately compile payloads. Lastly, a virtual peering network is needed to connect the two sub-networks.

### 3.2. Configuration

The process of configuring an environment varies a lot depending on the target operating system, and the attack simulation. Without a clear list of settings and software that need to be present, launching an attack is either unachievable, or produces inaccurate results. The countless ways a configuration setup can go amiss, and the needed technical knowledge and familiarity with the operating system make configuration a daunting task.

To configure the environment for the APT29 scenarios, one needs to connect to each resource through Windows Remote Desktop, an application that allows one to interact with a GUI for your virtual machines. The domain controller server needs to be setup by installing Active Directory (AD), creating a domain, adding the workstations to the domain along with creating a domain name service (DNS), group policy objects (GPO), domain users, and domain user groups. The workstations are then setup by installing additional software like Google Chrome, tampering with the registries and firewall rules, disabling Windows Defender, and ensuring that Windows Remote Management (WinRM) as well as other communication protocols and services are functioning correctly. Finally, on a command and control server (C2), one needs to install penetration testing tools such as Metasploit [7] to have a platform on which to launch the attack from.

The measures taken on the domain controller, workstations, and C2 are quite common for threat hunting cybersecurity research. Additionally, APT29 also requires the Powershell execution policy set to "Bypass", the registry modified to allow storage of wdigest credentials, the firewall configured to allow SMB (Server Message Block),

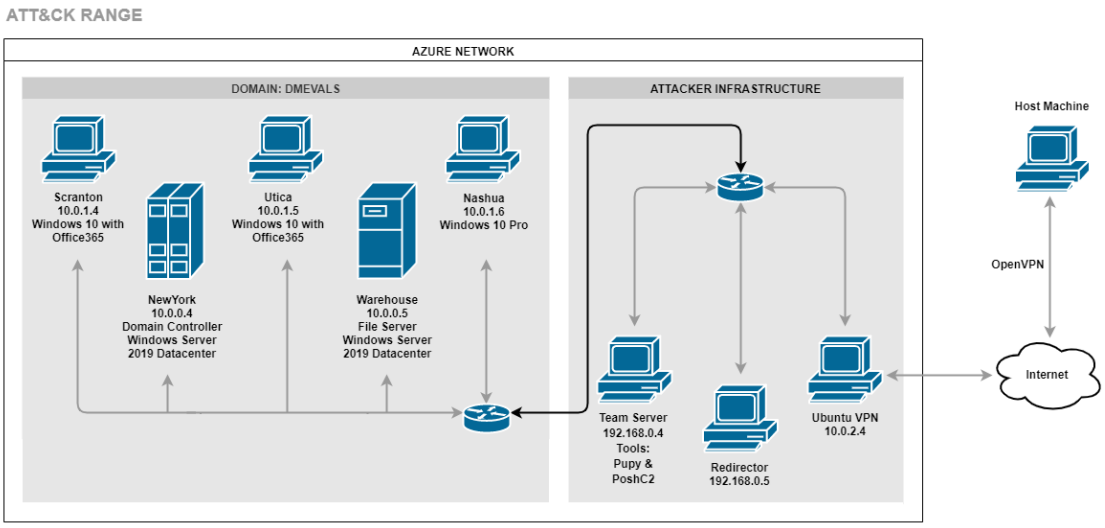


**Figure 1:** Schema of the APT29 Network

a SMB share present and working, the UAC (User Access Control) set to never notify for all Windows hosts.

Whereas the manual provisioning of the infrastructure was all initiated from a centralized cloud provider web platform with a user friendly interface, the manual configuration of an environment requires configuring the environment by connecting to different virtual machines endpoints. That framework offers a less controlled environment where the user has a harder time tracking the current state of configuration process and the remaining steps.

### 3.3. Attack Simulation

Once the infrastructure is deployed and configured, a researcher can then proceed to launch attacks. Every attack simulation requires an entry point from which malicious commands are executed and payloads downloaded. The entry point is an agent that lays in wait and listens for commands to execute from the C2 server. Therefore, one needs to connect to at least one workstation, and initiate the agent process before starting the attack simulation. The use of already infected workstations as a starting point for conducting post-breach cybersecurity research is common under the paradigm known as "Assume Breach Paradigm" [8]. Microsoft's Cyber Defense Operations Center describes it as such: "despite all the protections in place, we assume systems will fail or people will make errors, and an adversary may penetrate our infrastructure and services."

Once the workstation(s) are infected and the penetration testing tools ready, the attack simulation can be initiated. This step requires general knowledge about what the attack is trying to achieve, how it accomplishes its goals, what each step of the attack performs, also known as 'Techniques, Tactics and Procedures' (TTP), as well as more in-depth knowledge about how to navigate and send commands from the penetration testing tool.

The APT29 attack simulation is broken down into 2 different scenarios in order to depict the two approaches that the hacking group could deploy when they attack their targets. For both scenarios, the attacker uses a mix of Metasploit and Pupy in order to communicate with infected workstations and send shell commands to carry out the attacks. The first APT29 scenario represents a more aggressive, fast-paced, direct style that 'smashes and grabs' in order to reach its goals. The goal is to firstly collect and exfiltrate data, the focus then shifts to persistence, data collection, credential access, and lateral movement. The second scenario on the other hand is a stealthier and slower attack that looks at "establishing persistence, harvesting credentials, then finally enumerating and compromising the entire domain".[9]

More details about each step of the attack for both scenario 1 and 2 can be found in the appendix, where notes gathered from MITRE's adversary_emulation_library GitHub repository[9] can be found.

## 4. Automated security research

After studying how current manual security research is directed, we propose a way to conduct automated security research, and how to implement each of its steps. The automated solution is split in 3 main steps: provisioning (4.1), configuration (4.2), and attack simulation (4.3) as shown on figure 2. Additionally, data collection and reporting will be covered in 4.4, and a comparison between manual and automatic cyber security research will be drawn in 4.5.

One important and pressing issue reoccurring in the last section was the unavoidable complexity, and subsequently the required technical know-how one needs in order to carry out security research.This section is now going to see how to abstract out the intricacies of deploying, and configuring our cyber range through the concept of Infrastructure as Code (IaC) [10, 11]. As for attack simulations, the required technical knowledge that comes with penetration testing software is abstracted out through Caldera's intuitive GUI for managing agents, adversarial profiles, and operations.

Furthermore the automation technologies outlined for infrastructure provisioning and configuring have important attributes that make them all the more fit for cybersecurity research and development. The agentless nature of the technologies is an important step towards tackling the automation challenge as it avoids any unnecessary dependencies, limits the requirements to initiate the automation process, and minimizes the probability of critical errors. Moreover, the declarative capabilities of those technologies allow users to rapidly learn about and understand the different components that make up the automation process with little to no technical knowledge of the tools used. By separating the user from software implementation issues and edge case problems, a declarative style approach requires very little effort to build functional and robust programs.

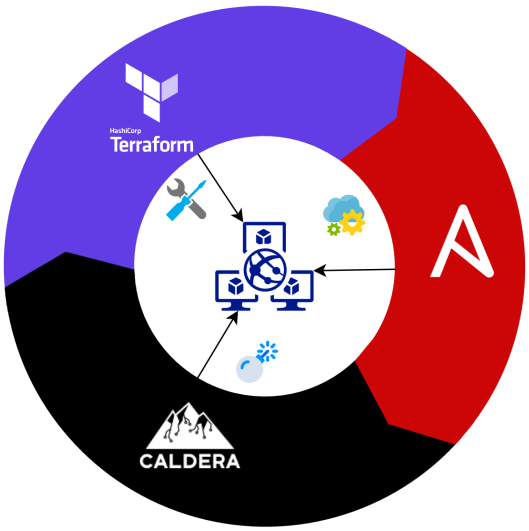


**Figure 2:** The 3 step automation cycle for cybersecurity research: Terraform, Ansible, and Caldera.

### 4.1. Provisioning

In order to provision our environment in a fully automated manner, we use Terraform [12], an open source IaC software tool. The tool allows for the creation and provisioning of infrastructure using HashiCorp Configuration Language (HCL), a declarative configuration language where the user 'declares' or writes in HCL code the desired state for the infrastructure.

Terraform offers the ability to write reusable and modular code. The re-usability feature gives Terraform a sizeable advantage over deploying each resource manually on a Cloud Provider. Users save on the amount of time the infrastructure takes to deploy: Terraform creates a resource dependency graph to set the order in which resources need to be deployed, it then instantly and concurrently creates infrastructure in a way that respects the dependency graph, and in a manner that humans could not compete with. But the leading gain is in the reduced amount of workload and time someone has to spend in order to boot up the infrastructure. The whole infrastructure can be built from 2 simple commands:

```
terraform plan -out {name_of_plan}
terraform apply {name_of_plan}
```

The user first creates a 'plan' that represents the changes Terraform counts on implementing such as destroying or creating a virtual machine, and then applies the planned changes. Finally, "terraform state list" shows the current state of the infrastructure. The user then doesn't have to spend 1 to 2 hours creating the resources by hand, but can spend that time on higher added value work whilst waiting for Terraform to complete. The modular nature of Terraform means that we can share our code, or a portion of it for others to reuse. There are no manual equivalents when one looks for a way to share the ability to launch an identical infrastructure, and show the desired infrastructure end-state. The code format of our infrastructure deployment, or IaC, enables us to use features that come with version controller systems (VCS) hosting platforms like GitHub or GitLab. It allows us to perform code reviews, work on particular branches of the code, look at history graphs, track issues and goals, and more broadly work in a collaborative, structured and fast-paced environment. Finally, in contrast to an imperative approach, the user does not need to know how Terraform implements HCL code and deploys the

infrastructure. Subsequently, a great deal of complexity is taken out of the hands of the user given that HCL is easy to read, learn and execute.

## 4.2. Configuration

As for the automation of configuration for our environment, Ansible [13], another open source IaC software tool released in 2012, and developed by Red Hat Inc since 2015, provides a wide range of packages and functions that allow for configuration management. Ansible uses a hybrid between imperative and declarative style language where development should be as declarative as possible but might still require imperative style code.

Ansible's modular programming capabilities are just as pronounced as Terraform's, not only are functions used within a source code file reusable, but so are larger and abstract goals such as setting up and configuring Windows' Active Directory can be packaged and encapsulated in a single package or 'role'. A role is a folder or package that contains both what the user might conceive as the main source code (tasks), as well as additional resources such as template files, variable files, handler files (files that manage exceptions and special conditions). They can then easily be shared with the community or within a workspace. Ansible's flexibility becomes important when looking at requirements and interoperability. Its requirements are minimal, as it only requires for Python to be installed, and for a mean of connection (WinRM or ssh [14]) to be available. Like Terraform, Ansible's use of IaC allows for code reviews and other perks, as well as being easy and quick to launch.

## 4.3. Attack simulation platform

There are various challenges with automatically measuring aspects of a network's security posture through penetration testing, red teams, and adversary emulation and numerous way to go about implementing it [15]. Cyber Adversary Language and Detection Engine for Red team Automation (CALDERA) version 3 [16] is a simulated penetration testing platform for autonomous red teaming. It is an open source tool developed by MITRE. CALDERA offers three major advantages over a manual style attack.

The first is that CALDERA has an interactive, friendly and graphical user interface, launching an attack is an intuitive and short procedure. The program has 3 main categories: Agents, Adversaries, and Operations. One can easily toggle from one to the other without any interference between each other. The 'Agents' section provides a dashboard with a list of agents currently running, their information and whether they have been terminated, as well as code to implant agents on a target workstation before one can simulate an attack. The 'Adversaries' section provides an interface to inspect the collection of abilities that make up an adversary, or an adversarial attack. One can either analyze the attack by overviewing the different abilities and getting a general understanding of how the attack works, or dig deeper in each ability, and inspect the commands launched. Lastly, the 'operations' section is at the core of launching attacks, it allows for the configuration and management of operations, with the capability to manually add and execute commands to an on-going autonomous attack.

The second advantage CALDERA has to offer is a platform on which one can easily build variations out of a particular adversarial attack, as well as producing, experimenting and sharing new adversaries. This feature is of upmost importance as it comprehensively captures the merit and spectrum of benefits that an automated system provides. It provides an unparalleled flexible structure to produce modular and automated attacks.

The third advantage CALDERA brings is a catalogue of attacks belonging to different threat actors, including the APT29 scenario. The utility for autonomous red teaming grows with the complexity of the simulated attacks, and given that the attacks available are of APT level sophistication, CALDERA becomes an invaluable tool. An advanced persistent threat (APT) is a threat group, usually associated with nation states, with advanced capabilities to penetrate systems and networks.

## 4.4. Data collection and reporting

The data collection process is specific to an operating system, but can be setup automatically at the configuration step through Ansible. Taking a deeper look at the data collection process for a windows ecosystem, a way to collect logs is through Windows Event Forwarding (WEF) [17], a service that comes as part of Windows 10, and allows workstations to forward local logs to other Windows machines. The main steps in setting up WEF can be split between setting up the workstations, also known as WEF clients, on which the attacks are unfolding, and the server listening for incoming connections from the WEF Clients. These steps include enabling the WinRM service, changing registry keys, changing audit policies and system access controls, and uploading XML files to configure the WEF service, all steps that can be completed automatically through Ansible packages and functions. There are plenty of services and products that take care of the data reporting process, and are usually chosen depending on the technology a user is most familiar with, or already has setup.

## 4.5. Results

The time results comparing a manual and automated approach to simulating the APT29 attack plans can be

| | Manual | Automatic |
|---|---|---|
| Provisioning | 1 - 2h | 17m |
| Configuring | 5 - 10h | 30m |
| Attack | 10 - 15h (30m if familiar) | 8m |
| Total | 6.5 - 27h | 55m |

**Table 1**
Time comparisons between manually and automatically completing APT29's scenario 1

found above in Table 1 (1). All in all, the deployment of the infrastructure, configuration of the environment, and completion of the attack simulation for the first scenario of APT29 takes just under an hour from start to finish. In contrast, for a cybersecurity researcher new to APT29 attacks, the simulation would likely take between 6.5 and 27 hours.

Using Terraform to deploy the infrastructure took 17 minutes, whilst deploying all of the infrastructure manually through Azure would take an amount of time in the scale of multiple hours. Configuration automation allows researchers to have a environment ready in 30 minutes. Configuring automatically with Ansible, here, allows us to save time on a process that would usually take between 5 - 10 hours.

Manually running an attack simulation is different from deploying an infrastructure or configuring it, in the sense that once an attack is mastered by a researcher, he can complete the attack in the same time order as he automatically would with an automated tool. CALDERA took a total of 8 minutes to run scenario 1 of the APT29 simulation, whilst a well versed researcher could finish it in less than 30 minutes.

The automation of a cyber range does not come without a price, cyber range automation allows us to accelerate the research development cycle but in turn takes away from potential expertise and knowledge that researchers would have developed in the process of creating a cyber range themselves. In the deployment and configuration step, the time savings justifies the expertise delegated to the automated platform. In contrast, spending time in the attack simulation stage to understand an attack and manually launch attack simulations is an important component to preserve in an automated cyber range. If expertise was delegated in the deployment and configuration stage, it is for researchers to spend more time on mastering the attack simulation stage. Nevertheless, automated red teaming presents an alternative and interesting way of conducting attack simulations. Whilst manually executing an attack simulation requires a grasp of every step before completing an attack, automated red teaming allows researchers to run full attack simulations without understanding certain steps, which is useful for understanding the general operational flow of an attack.

# 5. Relevance of automation for AI

Generating and updating a collection of diverse datasets is especially important in the cybersecurity field where threats, actors and their representations are constantly changing, and where experts have to be persistently learning about new paradigms, heuristics and technologies. The automatic construction of a cyber range presented in this paper does not just provide solutions for current threats, but a general framework in which one can continually conduct research, and build new datasets.

In AI development, results can only be as good as the quality of the data used. Therefore, having a limited amount of datasets to train good models and counting on them to protect organisations is not a viable solution. A fully automated cyber range ensures us to have access to diverse datasets. Its flexibility enables us to add variations on an attack, and create a wide array of different environments against which to generate datasets.

In this section we will discuss how such a framework does not just enhance cybersecurity researchers but also provide the necessary sandbox for AI researchers to develop applications and train models of good quality. We will firstly see how such a cyber range can help build a high level ontology in the domain of cybersecurity in 5.1. We then look at some commons pitfalls machine learning models encounter with poor data and how an automated cyber range can help us avoid them in 5.2.

## 5.1. Ontologies

"MITRE ATT&CK is a globally-accessible knowledge base of adversary tactics and techniques based on real-world observations". The tactics and procedures provided by the framework allow us to paint a meaningful picture for attacks [18]. If the MITRE ATT&CK framework provides the tools to build high level view attacks, and an automated system the low level log datasets of attack simulations (see figure 3), then Ontology [19] is the key to bridging the gaps between the two. It enables us to map thousands of logs into a coherent and comprehensible sequence of MITRE ATT&CK techniques, tactics and procedures.

Ontology-based data access (OBDA) is a well established paradigm for querying incomplete and heterogeneous data sources while incorporating knowledge from a domain ontology [20, 21]. OBDA allows a user to formulate queries through a high-level ontology vocabulary, delegating to the algorithm the task of querying low level data and mapping them back to high level concepts.

The ontological process represents the abstraction exercise that cybersecurity experts perform each day when looking at a collection of security logs, whether it is in an incident response, troubleshooting, or active monitoring context. An ontology must consequently define semantic

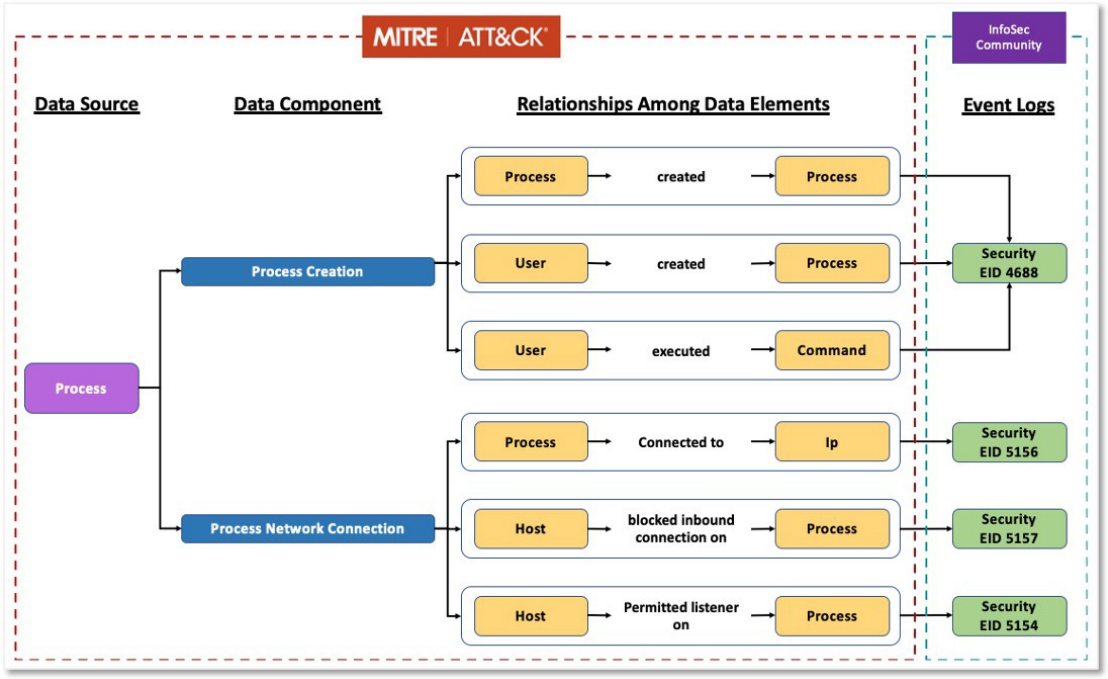


**Figure 3:** ATT&CK Data Sources (Defining ATT&CK Data Sources, Part I: Enhancing the Current State).

concepts that cybersecurity analysts use and recognise in order to abstract out the logs.

The diversity of our data allows us to test the level of expressiveness an ontology has to offer. Building an ontology only with high level concepts in mind might not scale to real data mapping. On the other hand, overfitting concepts on a limited amount of datasets puts us at risk of not being able to generalize when the paradigm shifts ever so slightly. Therefore, ontology building is an iterative process which is best served by an flexible and automated cyber range that can reliably produce heterogeneous and realist as possible data.

## 5.2. Machine learning

After developing an ontology, one can train machine learning models that learn from the same semantic concepts that cybersecurity experts use, enabling them to intelligently interpret predictions, as opposed to trying to learn directly from datasets of innumerable logs. Allowing machine learning models to base themselves off of high level abstractions, it empower them to be much more robust both to overfitting problems, and adversarial examples. Firstly, it allows models to avoid overfitting on meaningless features [22, 23] such as learning that communication with a particular IP address presents a high attack risk, and should therefore be blocked. Secondly, the heavy use of abstraction allowed through the use of ontologies enables the models to limit the effects of adversarial artificial intelligence examples. Deep learning models for instance, are known to be highly vulnerable to adversarial examples[24]. Introducing very superficial changes to an input can make predictions highly unstable and inaccurate, a situation where humans can reliably understand that the input has not significantly changed.

Ultimately, the hope is for the machine learning models to grasp abstract concepts and general pattern recognition, and discover new heuristics for cybersecurity analysts to integrate in their practice. As can be seen on figure 4 (4), among the most pressing issues that cybersecurity analysts are trying to solve is the problem of overwhelming false positives (A2 and A3). The problem clouds analysts' judgment for potential threats, and causes what is known as 'Alert Fatigue', a fatigue produced by a myriad of false positives that continually drain analysts' attention. Other important issues pointed out in Panther Labs' survey findings (Figure 4) are a lack of context for alerts and insights given by current SIEM systems to experts, and the sheer number of those alerts (A1 and A4). Using an approach to machine learning based on ontology, the enrichment of our data from initial logs to high level data would allow models to put

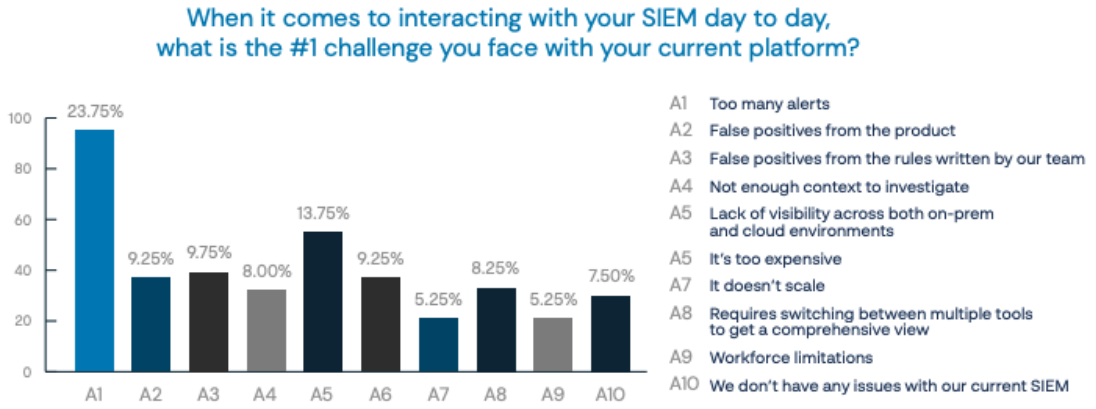


**Figure 4:** Panther Labs' cybersecurity survey on the current state of SIEM ( State of SIEM 2021 Insights From 400 Security Professionals )

in the hands of cybersecurity experts meaningful and contextual alerts.

## 6. Conclusion

The automation solution we brought forward in this paper is designed to help cybersecurity researchers looking to integrate AI in their operations, as well as AI researchers interested in contributing to the cybersecurity field. We used a scenario inspired from an APT29 attack campaign to better illustrate the benefits that the automation platform brings for researchers. The solution enables researchers to operate and build software on top of an automated cyber range, allowing them to save time and focus solely on the development of artificial intelligence tools. Terraform automates the deployment of the infrastructure, Ansible automates its configuration, Caldera provides autonomous red teaming capabilities for attack simulations, and WEF helps centralize and collect the attack simulation's data.

## References

[1] Common staff target for military cooperation on cyber ranges in the european union, 2013. URL: https://eda.europa.eu/docs/default-source/procurement/annex-a---cyber-ranges-cst.pdf.

[2] Mitre engenuity website, 2019. URL: https://mitre-engenuity.org/.

[3] Center for threat informed defense website, 2019. URL: https://ctid.mitre-engenuity.org/.

[4] Mitre att&ck website, 2015. URL: https://attack.mitre.org/groups/G0016/.

[5] Attack range github repository, 2019. URL: https://github.com/splunk/attack_range.

[6] Simuland github repository, 2021. URL: https://github.com/Azure/SimuLand.

[7] D. Kennedy, J. O'gorman, D. Kearns, M. Aharoni, Metasploit: the penetration tester's guide, No Starch Press, 2011.

[8] R. Pompon, Assume Breach, Apress, Berkeley, CA, 2016, pp. 13–21. URL: https://doi.org/10.1007/978-1-4842-2140-2_2. doi:10.1007/978-1-4842-2140-2_2.

[9] Adversary emulation library github repository, 2019. URL: https://github.com/center-for-threat-informed-defense/adversary_emulation_library/tree/master/apt29/Emulation_Plan.

[10] A. Rahman, R. Mahdavi-Hezaveh, L. Williams, A systematic mapping study of infrastructure as code research, Information and Software Technology 108 (2019) 65–77. URL: https://www.sciencedirect.com/science/article/pii/S0950584918302507. doi:https://doi.org/10.1016/j.infsof.2018.12.004.

[11] C. Parnin, E. Helms, C. Atlee, H. Boughton, M. Ghattas, A. Glover, J. Holman, J. Micco, B. Murphy, T. Savor, M. Stumm, S. Whitaker, L. Williams, The top 10 adages in continuous deployment, IEEE Software 34 (2017) 86–95. doi:10.1109/MS.2017.86.

[12] Mitchell Hashimoto et al , Terraform website, 2014. URL: https://www.terraform.io/.

[13] Ansible website, 2012. URL: https://www.ansible.com/.

[14] T. Ylonen, C. Lonvick, et al., The secure shell (ssh) protocol architecture, 2006.

[15] J. Hoffmann, Simulated penetration testing: From

"dijkstra" to "turing test++", in: Proceedings of the Twenty-Fifth International Conference on International Conference on Automated Planning and Scheduling, ICAPS'15, AAAI Press, 2015, p. 364–372.

[16] Caldera, a scalable, automated adversary emulation platform, 2021. URL: https://caldera.mitre.org/.

[17] Spotting the Adversary with Windows Event Log Monitoring, Technical Report, NSA, 2015.

[18] Best Practices for MITRE ATT&CK Mapping, Technical Report, CISA, HSSEDI, 2021.

[19] D. L. McGuinness, F. Van Harmelen, et al., Owl web ontology language overview, W3C recommendation 10 (2004) 2004.

[20] M.-L. Mugnier, M.-C. Rousset, F. Ulliana, Ontology-Mediated Queries for NOSQL Databases, in: DL: Description Logics, volume CEUR Workshop Proceedings, Cape Town, South Africa, 2016, pp. 1051–1057. URL: https://hal-lirmm.ccsd.cnrs.fr/lirmm-01375093, this paper is an extended abstract of the paper with the same title presented at AAAI 2016.

[21] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, R. Rosati, Linking data to ontologies, in: S. Spaccapietra (Ed.), Journal on Data Semantics X, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 133–173.

[22] J. Reunanen, Overfitting in making comparisons between variable selection methods, J. Mach. Learn. Res. 3 (2003) 1371–1382.

[23] R. B. Rao, G. Fung, R. Rosales, On the Dangers of Cross-Validation. An Experimental Evaluation, Society for Industrial and Applied Mathematics, 2008, pp. 588–596. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611972788.54. doi:10.1137/1.9781611972788.54.

[24] P.-A. Moëllic, The dark side of neural networks: an advocacy for security in machine learning, Computer & Electronics Security Applications Rendezvous (C&ESAR) (2018).

# Appendix

Details about each step of the APT29 attack simulation for both scenario 1 and 2 are compiled here, as referenced in 3.3. The details were gathered from MITRE's adversary_emulation_library GitHub repository[9].

## Scenario 1

The scenario begins with an initial breach, where a legitimate user clicks (T1204 / T1204.002) an executable payload (screensaver executable) masquerading as a benign word document (T1036 / T1036.002). Once executed, the payload creates a C2 connection over port 1234 (T1065) using the RC4 cryptographic cipher. The attacker then uses the active C2 connection to spawn interactive cmd.exe (T1059 / T1059.003) and powershell.exe (T1086 / T1059.001).

The attacker runs a one-liner command to search the filesystem for document and media files (T1083, T1119), collecting (T1005) and compressing (T1002 / T1560.001) content into a single file. The file is then exfiltrated over the existing C2 connection (T1041). The attacker now uploads a new payload (T1105) to the victim. The payload is a legitimately formed image file with a concealed PowerShell script (T1027 / T1027.003). The attacker then elevates privileges via a user account control (UAC) bypass (T1122 / T1546.015, T1088 / T1548.002), which executes the newly added payload. A new C2 connection is established over port 443 (T1043 using the HTTPS protocol (T1071 / T1071.001, T1032 / T1573). Finally, the attacker removes artifacts of the privilege escalation from the Registry (T1112).

The attacker uploads additional tools (T1105) through the new, elevated access before spawning an interactive powershell.exe shell (T1086 / T1059.001). The additional tools are decompressed (T1140) and positioned on the target for usage. The attacker then enumerates running processes (T1057) to discover/terminate the initial access from Step 1 before deleting various files (T1107 / T1070.004) associated with that access. Finally, the attacker launches a PowerShell script that performs a wide variety of reconnaissance commands (T1016, T1033, T1063 / T1518.001, T1069, T1082, T1083), some of which are done by accessing the Windows API (T1106).

The attacker establishes two distinct means of persistent access to the victim by creating a new service (T1031 / T1543.003) and creating a malicious payload in the Windows Startup folder (T1060 / T1547.001). The attacker collects screenshots (T1113), data from the user's clipboard (T1115), and keystrokes (T1056 / T1056.001). The attacker then collects files (T1005), which are compressed and encrypted (T1560 / T1560.001), before being exfiltrated to an attacker-controlled WebDAV share (T1048 / T1048). The attacker uses Lightweight Directory Access Protocol (LDAP) queries to enumerate other hosts in the domain (T1018) before creating a remote PowerShell session to a secondary victim (T1021 / T1021.006). Through this connection, the attacker enumerates running processes (T1057). Next, the attacker uploads (T1105) a new UPX-packed payload (T1027 / T1027.002) to the secondary victim. This new payload is executed on the secondary victim via the PSExec utility (T1021 / T1021.002, T1035 / T1569.002) using the previously stolen credentials (T1078 / T1078.002).

The attacker uploads additional utilities to the secondary victim (T1105) before running a PowerShell one-liner command (T1059 / T1059.001) to search for filesystem for document and media files (T1083, T1119). Files of interested are collected (T1005) then encrypted and compressed (T1002, T1022 / T1560.001 into a single file (T1074 / T1074.001). The file this then exfiltrated over the existing C2 connection (T1041). Finally, the attacker deletes various files (T1107 / T1070.004) associated with that access.

The original victim is rebooted and the legitimate user logs in, emulating ordinary usage and a passage of time. This activity triggers the previously established persistence mechanisms, namely the execution of the new service (T1035 / T1569.002) and payload in the Windows Startup folder (T1060 / T1547.001). The payload in the Startup folder executes a follow-on payload using a stolen token (T1106, T1134 / T1134.002).

## Scenario 2

The scenario begins with initial breach, where a legitimate user clicks (T1204 / T1204.002) a link file payload, which executes an alternate data stream (ADS) hidden on another dummy file (T1096 / T1564.004) delivered as part of the spearphishing campaign. The ADS performs a series of enumeration commands to ensure it is not executing in a virtualized analysis environment (T1497 / T1497.001, T1082, T1120, T1033, T1016, T1057, T1083) before establishing persistence via a Windows Registry Run key entry (T1060 / T1547.001) pointing to an embedded DLL payload that was decoded and dropped to disk (T1140). The ADS then executes a PowerShell stager (T1086 / T1059.001) which creates a C2 connection over port 443 (T1043) using the HTTPS protocol (T1032 / T1573.002 , T1071 / T1071.001).

The attacker modifies the time attributes of the DLL payload (T1099 / T1070.006) used in the previously established persistence mechanism to match that of a random file found in the victim's System32 directory (T1083). The attacker then enumerates registered AV products (T1063 / T1518.001) and software installed by the user documented in the Windows Registry (T1012).

The attacker performs local enumeration using various Windows API calls, specifically gathering the local

computer name (T1082), domain name (T1016), current user context (T1033), and running processes (T1057).

The attacker elevates privileges via a user account control (UAC) bypass (T1122 / T1546.015, T1088 / T1548.002). The attacker then uses the new elevated access to create and execute code within a custom WMI class (T1047) that downloads (T1105) and executes Mimikatz to dump plaintext credentials (T1003 / T1003.001), which are parsed, encoded, and stored in the WMI class (T1027). After tracking that the WMI execution has completed (T1057), the attacker reads the plaintext credentials stored within the WMI class (T1140).

The attacker establishes a secondary means of persistent access to the victim by creating a WMI event subscription (T1084 / T1546.003) to execute a PowerShell payload whenever the current user (T1033) logs in.

The attacker enumerates the environment's domain controller (T1018) and the domain's security identifier (SID) (T1033) via the Windows API (T1106). Next, the attacker uses the previously dumped credentials (T1078 / T1078.002) to create a remote PowerShell session to the domain controller (T1028 / T1021.006). Through this connection, the attacker copies the Mimikatz binary used in Step 14 to the domain controller (T1105 / T1570) then dumps the hash of the KRBTGT account (T1003 / T1003.001).

The attacker harvests emails stored in the local email client (T1114 / T1114.001) before collecting (T1005) and staging (T1074 / T1074.001) a file of interest. The staged file is compressed (T1002 / T1560.001) as well as prepended with the magic bytes of the GIF file type (T1027).

The attacker maps a local drive to an online web service account (T1102) then exfiltrates the previous staged data to this repository (T1048 / T1567.002).

The attacker deletes various files (T1107 / T1070.004) associated with that access by reflectively loading and executing the Sdelete binary (T1055 / T1055.002) within powershell.exe.

The original victim is rebooted and the legitimate user logs in, emulating ordinary usage and a passage of time. This activity triggers the previously established persistence mechanisms, namely the execution of the DLL payload (T1085 / T1218.011), referenced by the Windows Registry Run key, and the WMI event subscription (T1084 / T1546.003), which executes a new PowerShell stager (T1086 / T1059.001). The attacker uses the renewed access to generate a Kerberos Golden Ticket (T1097 / T1558.001, T1558.003), using materials from the earlier breach, which is used to establish a remote PowerShell session to a new victim (T1028 / T1021.006). Through this connection, the attacker creates a new account within the domain (T1136 / T1136.001).