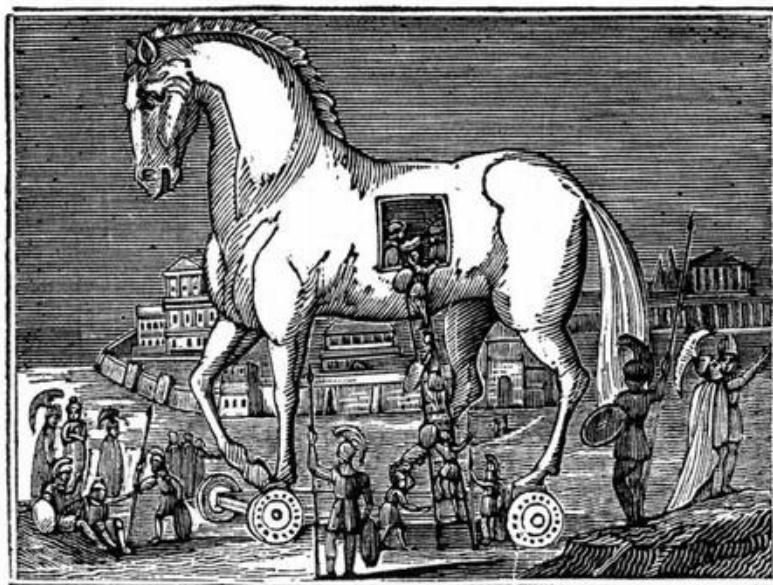


VERAMINE DYNAMIC DECEPTION SYSTEM

Version 1.8.2



Trojans Deceived.

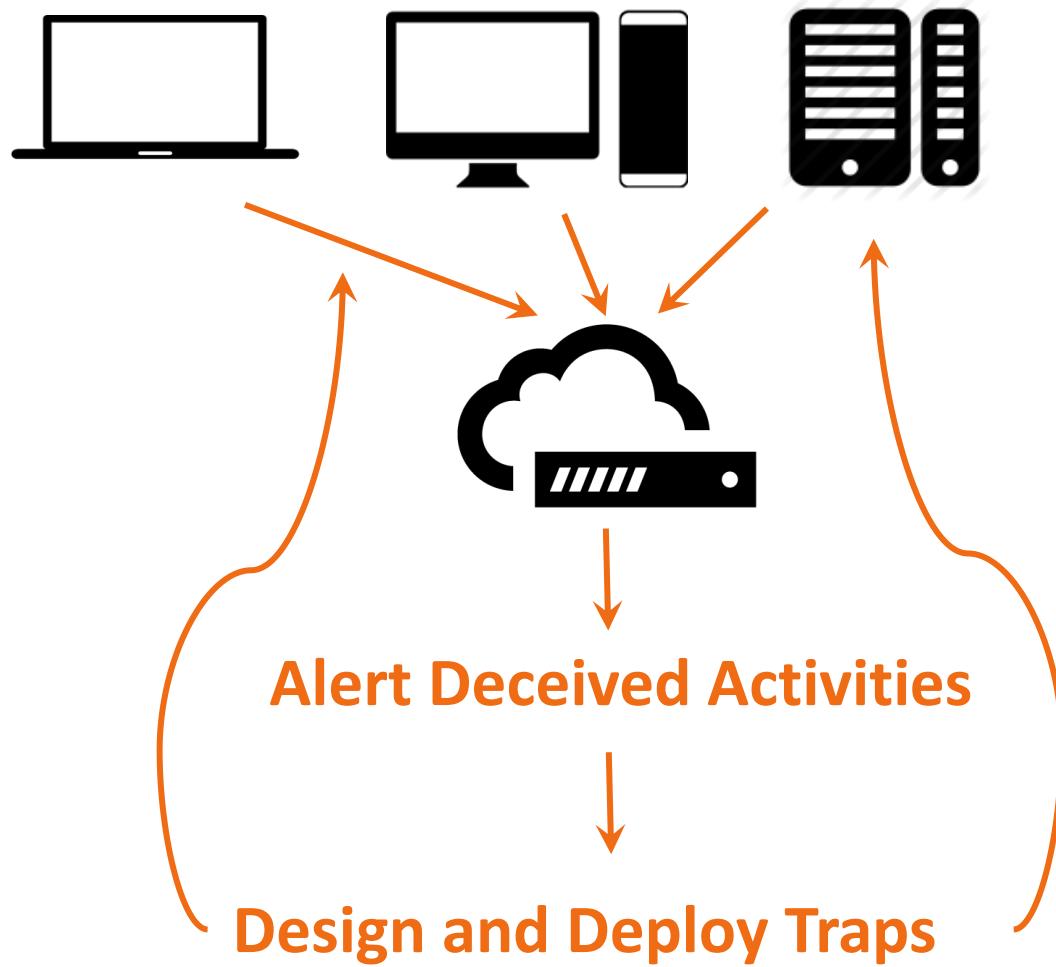
AUGUST 21, 2019

Mục lục

Deception Introduction.....	3
Deployment procedures	4
Tactic description and parameters	6
Services	6
Processes.....	7
Files	8
Mutexes	8
Events.....	9
Network listener	10
Credentials	10
Network shares.....	11
Registry	13
Appendix A: Outstanding Features of Veramine Suite	14
Data Collection and Advanced Monitoring.....	14
Detection and Deception.....	14
Incident Response and Forensics.....	15
Performance, Deployment, Integration and Management	15

Deception Introduction

Dynamic Deception System (DDS), a Platform of Security Traps uniquely offered by Veramine, such as Deceptive services, processes, mutexes, credentials, network listeners, data shares, registry helper..., that can be deployed to any set of Hosts, as an Active Defense approach to Detect and Prevent attacks. Most other existing approaches are Passive Defense. VDDS is capable of making every computer (physical or VM) a honeypot, in IT Systems. The traps are put along the kill chain, to cheat, detect and prevent intrusions, track intruders' activities, and limit things they can do.



The Veramine platform provides a set of deception tactics that can be customized and deployed to endpoints. A tactic is a type of deception, i.e., a file, a service, mutex, etc. These files backing these tactics are controlled by Veramine. The supported tactics are:

1. Services
2. Processes
3. Files
4. Registry
5. User credentials

6. SMB shares
7. Network listeners
8. Mutexes
9. Events

The parameters for each tactic are described in a later section.

Deployment procedures

Tactics are listed and updated using JSON via the portal (Settings, Policies, Deception):

The screenshot shows the Veramine portal interface. On the left is a dark sidebar with various navigation options: Services, Monitored Hosts, Tracked Users, Binaries, Detections, Response, Search, and Settings. The 'Settings' option is currently selected. The main content area has a header with 'Manage deception policies' and a search bar. Below this, there's a list of 'Saved Policies' including 'Fake Admin Share', 'Fake Credentials', 'Fake Listener', 'Fake Mutexes', 'Fake Process', 'Fake Service', and 'Fake VirtualBox'. A 'New Policy' button is located at the bottom of this list. To the right, a detailed view of the 'Fake Admin Share' policy is shown in a modal window. The window title is 'Fake Admin Share' and it describes it as a 'Deception Policy that creates a share with fake sensitive files'. Below this is a 'Policy Settings (JSON)' section containing the following JSON code:

```
{
  "files": [
    {
      "fileName": "passwords.xlsx",
      "subPath": "adminshare",
      "baseDirectory": "deception"
    },
    {
      "fileName": "cert.pem",
      "subPath": "adminshare",
      "baseDirectory": "deception"
    }
  ]
}
```

At the bottom of this section is a link 'Edit policy definition in JSON format'. At the very bottom of the page is a blue 'Update' button.

Deception policy setting

There are pre-made deception tactics which can be deployed immediately. If you want to create a new deception tactic, click “New Policy” to create a new deception policy; fill in the information as requested and click “Create”,

The screenshot shows the Veramine interface with a sidebar on the left containing links like Monitored Hosts, Tracked Users, Binaries, Detections, Response, Search, and Settings. The main area is titled "Create new Deception policy". It has fields for "Name" (containing "test deception policy") and "Description" (containing "test deception campaign"). Below these is a "Short description of a new policy" field. At the bottom right are "Cancel" and "Create" buttons. To the right of the "Create" button is a JSON editor with the following content:

```

{
  "name": "test deception policy",
  "description": "test deception campaign",
  "shortDescription": "Short description of a new policy",
  "tactics": [
    {
      "id": "T1059",
      "name": "File Deletion (Windows)",
      "parameters": [
        {
          "name": "filename",
          "value": "testSvc.exe"
        }
      ]
    }
  ],
  "tags": []
}

```

Below the JSON editor is a link "Edit policy definition in JSON form".

Deception policy box

Policies can also be updated by changing the JSON and clicking “Update” at the bottom right.

To deploy a tactic, go to the “Groups” view, right click on the group and select “Configure Policies”,

The screenshot shows the "Monitored Hosts" view with a header "Monitored Hosts (Displaying hosts on the network)". Below it is a table with columns: Group, # Hosts, Sensor Status, Version, Collection Policy, and Deployment. The "Default" group is selected. A context menu is open over the "Default" group, listing the following options:

- Configure Policies
- Set Auto-assignment Criteria
- Set Sensor State
- Rename
- Deploy Sensor Update
- Delete

At the bottom of the screen, there are status indicators: "Status: Normal" and "© 6:06:13am (UTC), Jul 2, 2018".

Configure policies for a group

Then you can pick the deception policy from the list and click “Update.”

The screenshot shows a 'Configure policies' dialog box. On the left, there's a sidebar with metrics like 'Online hosts' (0 out of 2000), 'Offline hosts' (2000), and sections for 'Status', 'Platform', and 'OS Sec'. The main area has fields for 'Group name' (Default) and 'Host count' (1959). Below these is a note about configuration policies being pushed to all machines in the group. A 'Collection policy' dropdown is shown, and a 'Deception policy' dropdown is open, displaying options: None, Fake Admin Share, Fake Credentials, Fake Listener, Fake Mutexes, **Fake Process**, Fake Service, and Fake VirtualBox. At the bottom are 'Cancel' and 'Update' buttons.

Deploying policies to a group

To uninstall a policy, simply change it to “None” and click update.

Tactic description and parameters

The following section describes each tactic and its parameters.

Services

You can deploy a service with a specified filename on the remote system. The parameters are:

- **name**: the service’s name (i.e., the parameter to `sc stop/start`). It cannot have spaces.
- **displayName**: the service’s long description.
- **filename**: the name to be used for the service executable.

Here is a JSON describing two services,

```
{  
  "services": [  
    {  
      "name": "testSvc",  
      "displayName": "test display name",  
      "fileName": "testSvc.exe"  
    },  
    {  
      "name": "testSvc2",  
      "displayName": "test2 display name",  
      "fileName": "testSvc2.exe"  
    }  
  ]  
}
```

Example of a deceptive service is tampered with:

Risk	Rece... ↴	Description	Dt	Hosts
Medium	8/31/17 4:17:54pm	A user terminated a deception process on host pc06	▼ ...	pc06.verami...
Medium	8/31/17 4:17:53pm	A user tampered with a deception service on host pc06.	▼ ...	pc06.verami...
Time ↴	Description			
8/31/17 4:17:53pm	Deception Service SecMonSvc on host pc06 was modified, possibly by an attacker. The configuration changed from (85) to (69)			

Processes

You can specify a process with a certain name to be launched. The parameters are:

```
{  
  "processes": [  
    {  
      "fileName": "process.exe",  
      "baseDirectory": "ProgramFiles"  
    }  
  ]  
}
```

The valid base directories include the following: [ProgramFiles](#), [ProgramFilesX86](#), [Windows](#), [System32](#), [Drivers](#), [Deception](#).

Example of a deceptive process is terminated:

Risk	Rec...	Description	Dt	Hosts
Medium	8/31/17 4:07:02...	A user terminated a deception process on host pc06	...	pc06.verami...
Time	Description			
8/31/17 4:07:02pm	Deception Process secmonitor.exe was unexpectedly terminated with exitcode 4294967295, possibly by an attacker.			

Files

Files can be placed in specific directories. The parameters are:

- **fileName**: the file name. The content of the file is provided by Veramine.
- **baseDirectory**: the base directory in which to write the file.

Example:

```
{
  "files": [
    {
      "fileName": "test1.exe",
      "baseDirectory": "Windows"
    },
    {
      "fileName": "test2.exe",
      "baseDirectory": "Windows"
    }
  ]
}
```

Mutexes

You can create a mutex. The parameters are:

- **name**: the name of the mutex. If you want it to be global, it needs to be prefixed with "Global\".

```
{
  "mutexes": [
    {
      "name": "Global\\crazyMutex"
    }
  ]
}
```

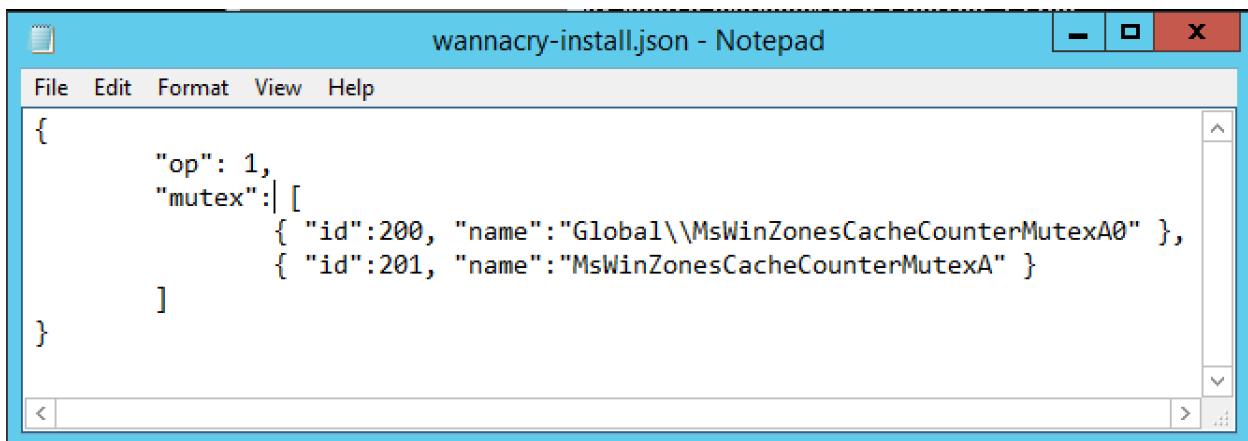
Example, WannaCry checks a mutex to decide if a system is already infected. We can set such a deceptive mutex.



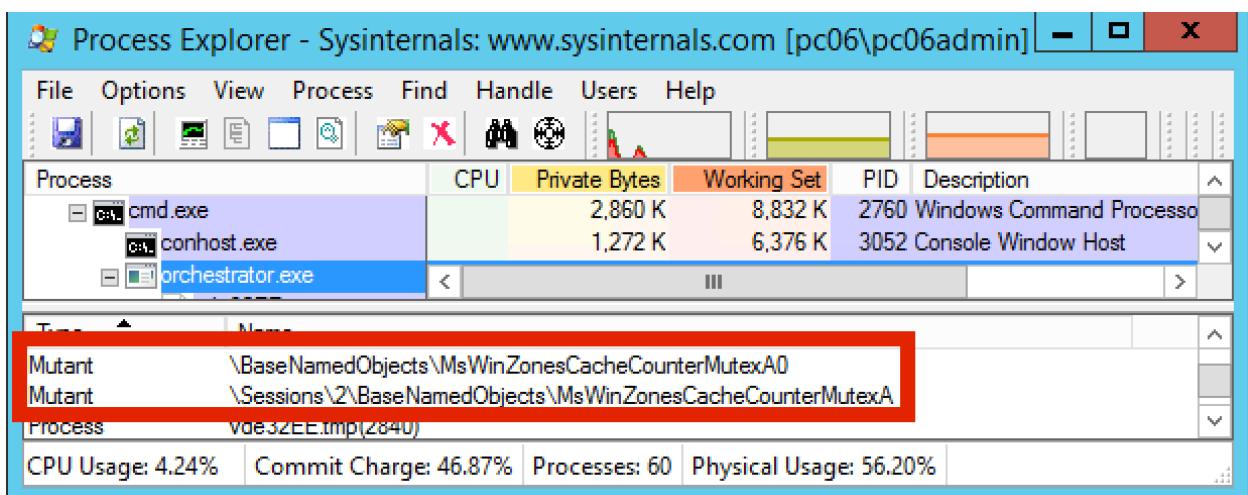
Florian Roth @cyb3rops · May 14

Another WannaCry vaccine:

Create a mutex named "**MsWinZonesCacheCounterMutexA**"



```
wannacry-install.json - Notepad
File Edit Format View Help
{
    "op": 1,
    "mutex": [
        { "id":200, "name":"Global\\MsWinZonesCacheCounterMutexA0" },
        { "id":201, "name":"MsWinZonesCacheCounterMutexA" }
    ]
}
```



Process Explorer - Sysinternals: www.sysinternals.com [pc06\pc06admin]

Process	CPU	Private Bytes	Working Set	PID	Description
cmd.exe		2,860 K	8,832 K	2760	Windows Command Processor
conhost.exe		1,272 K	6,376 K	3052	Console Window Host
orchestrator.exe					

Mutant \BaseNamedObjects\MsWinZonesCacheCounterMutexA0
Mutant \Sessions\2\BaseNamedObjects\MsWinZonesCacheCounterMutexA
Process vde32EE.tmp[2840]

CPU Usage: 4.24% Commit Charge: 46.87% Processes: 60 Physical Usage: 56.20%

Events

You can create an event. The parameters are:

- name: the name of the event. If you want it to be global, it needs to be prefixed with "Global\".

Example:

```
{
    "events": [
        {
            "name": "Global\\crazyEvent"
        }
    ]
}
```

```
]  
}
```

Network listener

A network listener is a process that binds to a TCP port and accepts connections. The parameters are:

- **port**: TCP port to listen on.

Example:

```
{  
  "listeners": [  
    {  
      "port": 31337  
    }  
  ]  
}
```

Medium	8/31/17 4:42:52...	A user connected to a deception network listener on host pc06.	<input type="button" value="▼"/> ...	pc06.verar...
Time	↓	Description	Host	
8/31/17 4:42:52pm		Network connection was made to port TCP/22 listening for deception purposes. Connection originated from remote 10.1.2.7:54921.	pc06	

Credentials

Credentials can be injected to Windows authentication subsystem. Note that these usernames/passwords may be real or fictitious. The parameters are:

- **domain**: the domain of the user.
- **userName**: username.
- **password**: password.

Example:

```
{  
  "credentials": [  
    {  
      "domain": "exampleDomain",  
      "username": "user",  
      "password": "myPassword"  
    }  
  ]  
}
```

```
]  
}
```

The screenshot shows a Notepad window titled "creds-install.json - Notepad". The file contains a JSON object with the following structure:

```
{  
    "op": 1,  
    "cred": [ { "id": 1, "username": "tempadmin1",  
               "domainName": "COMPANYNAME", "password": "P@s$w0rd1!1!1!" } ]  
}
```

A red box highlights the "password" field of the first credential entry.

The screenshot shows a terminal window titled "mimikatz 2.1.1 x64 (oe.eo)". The user has run the command "sekurlsa::logonpasswords". The output shows various credential entries, with a red box highlighting the Kerberos section:

```
mimikatz # privilege::debug  
Privilege '20' OK  
mimikatz # sekurlsa::logonpasswords  
  
Authentication Id : 0 ; 154212135 <00000000:09311727>  
Session : NewCredentials from 0  
User Name : pc06admin  
Domain : pc06  
Logon Server : <null>  
Logon Time : 9/1/2017 9:43:42 PM  
SID : S-1-5-21-2137512653-3017395921-3799018531-500  
msv :  
[00000003] Primary  
* Username : tempadmin1  
* Domain : COMPANYNAME  
* NTLM : d745b6768525c48159fb446cea183b92  
* SHA1 : 84c7b040fa2332a3ecd30e950abf90b15ab5848d  
tspkg :  
wdigest :  
* Username : tempadmin1  
* Domain : COMPANYNAME  
  
kerberos :  
* Username : tempadmin1  
* Domain : COMPANYNAME  
* Password : P@s$w0rd1!1!1!  
ssn :
```

Network shares

Network shares share a directory to the world. The parameters are:

- name: the name of the network share.
- description: the description of the network share.

Example:

```
{  
    "shares": [  
        {  
            "name": "testShare",  
            "description": "Test Share"  
        }  
    ]  
}
```

]
}

```
Command Prompt  
C:\Users\jness>net use \\pc06\tempadminshare  
The command completed successfully.  
  
C:\Users\jness>dir \\pc06\tempadminshare  
Volume in drive \\pc06\tempadminshare has no label.  
Volume Serial Number is EC44-7DBC  
  
Directory of \\pc06\tempadminshare  
08/31/2017 11:23 PM <DIR> .  
08/31/2017 11:23 PM <DIR> ..  
08/31/2017 11:23 PM 7 passwords.txt  
      1 File(s)     7 bytes  
      2 Dir(s) 125,253,251,072 bytes free  
  
C:\Users\jness>type \\pc06\tempadminshare\passwords.txt  
test  
C:\Users\jness>_
```

SHARES					
CONNECTIONS					
FILE ACCESS					
Status: Any	Type: Disk				
Status	Type	Share Name	Path	Owner	Create Time
Active	Disk	tempadminshare	c:\Windows\Phantom\deception\share1	BUILTIN\Administrators	8/31/17 4:23:59...
10	Showing all 1 entries				

SHARES					
CONNECTIONS					
FILE ACCESS					
Status: Active	Type: Any				
Status	Type	Share Name	Path	User	IP Address
Active	IPC	IPC\$		VERAMINE\jness	10.1.2.7
Active	Disk	tempadminshare	c:\Windows\Phantom\deception\share1	VERAMINE\jness	10.1.2.7
10	Showing all 2 entries				

SHARES				
CONNECTIONS				
FILE ACCESS				
Path	Access	User	IP Address	Time
passwords.txt	Read	VERAMINE\jness	10.1.2.7	8/31/17 4:32:02pm

Risk	Rece...	Description	Dt	Hosts	Inst #
Medium	8/31/17 4:25:12pm	A user established a connection to a deception share on host pc06.	<input type="button" value="^"/> ...	pc06.verami...	1
Time	Description			Host	
8/31/17 4:25:12pm	Deception SMB Share tempadminshare on host pc06 was accessed by user VERAMINE\jness from host pc06 with IP address(es) 10.1.2.7			pc06	

Risk	Rece...	Description	Dt	Hosts
Medium	8/31/17 4:32:02...	A user accessed a file on a deception share on host pc06.	<input type="button" value="^"/> ...	pc06.veran...
Time	Description			Host
8/31/17 4:32:02pm	A file named passwords.txt on Deception Share tempadminshare (host pc06) was opened for reading by user VERAMINE\jness from host with IP address(es) 10.1.2.7			pc06

Registry

Registry values can be set matching interesting preconfigured scenarios. The current version sets registry keys to spoof the presence of a virtual machine. The tactic names are:

- VMware: spoof VMware registry keys.
- VirtualBox: spoof VirtualBox registry keys.
- Qemu: spoof Qemu registry keys.
- HyperV: spoof HyperV registry keys.

Example:

```
{
  "reg": [
    {
      "tactic": "VMWare"
    }
  ]
}
```

Appendix A: Outstanding Features of Veramine Suite

Veramine Inc.

Advanced Endpoint Security

Specialized in building **cybersecurity endpoint** products, awarded contracts worth **multi-million USDs** from

- **U.S. Department of Homeland Security (DHS)**, also recommended by DHS as a platform for **financial and banking sector customers**
- **U.S. Department of Defense (DOD)**
- **U.S. Airforce**
- **ANZ, a top-3 bank in Australia**
- **And other important customers...**

Products, for **SOC, MSSP or IT admins, On-premise or Cloud**

- Veramine Endpoint Detection and Response (**VEDR**)
 - Veramine Dynamic Deception System (**VDDS**)
 - Veramine Advanced Activities Monitoring (**VAAM**)
-

Customers' **Compliments**: "unique and powerful capabilities for detailed data collection, monitoring, control, yara memory search, forensics, incident response, and detection"

Data Collection and Advanced Monitoring

Data Quality: Wide Variety. Detailed. Structured. Real Time. Small Traffic. All security-related activities, especially System Security and SMB data, is probably only collected by Veramine: **Process, Registry, System Security, Network, User, SMB, Binaries...**

Flexible collection policies: admins can select what data to collect. **Adaptive filter:** sensors smartly don't send irrelevant high-volume events to servers, that can filter out TB's of traffic sent and processed by sensors and servers.

External and Insider Threats Prevention with Advanced Monitoring on Data, Devices and Users, such as User and Entity Behavior Analytics (UEBA), Key loggers, Screenshot captures, Activities of Browsing-Email-SMB, User sessions, USB Management Logged Tracking and Access Control Policies (Blocked, Read-Only, or Read-Write).

Detection and Deception

Aim to detect all **attack tactics and techniques** in https://attack.mitre.org/wiki/Technique_Matrix, the Attack Dictionary.

More collected data types allow more data analysis algorithms, combining rule-based and machine learning, resulting in **better Detection**. Examples: SMB data allows detecting **Lateral Movement** and **Insider Threat**; Precise Elevation of Privilege (**EOP**) detection by collecting security tokens; Lsass process open allows detecting credentials and passwords dumping (**Mimikatz**); Command arguments allow detecting **Malicious Powershell Fileless** intrusion...

Deception is an **Active Defense** approach, whereas most existing approaches are **Passive Defense**. **Platform of Traps**, put along the kill chain, to **cheat, detect and prevent** intrusions. Capable of making **every computer** (physical or VM) a **honeypot**, in IT Systems. **Uniquely offered by Veramine**.

Deceptive services, processes, files, mutexes, credentials, network listeners, data shares, registry helper, VMs... Track intruders' activities, and limit things they can do, with the traps. E.g. WannaCry checks a mutex to decide if a system is already infected. We can set such a deceptive mutex.

Incident Response and Forensics

Yara Search on Memory and Files is Unique of Veramine. Memory dumps are at fingertips. All collected data is searchable using very flexible logical expressions. All executable binaries are collected for forensics.

Veramine have **most Response Actions**, from **Binaries, Users, Hosts to Processes**. E.g. Network Quarantine, Process Suspend/Terminate, User Disable/Disconnect, Host Sleep/Shutdown/Restart, Binary Block, Scan with Virus Total...

Forensics with Velociraptor to collect various **built-in or customized artifacts** from **multiple endpoints in real-time** from centralized portal. **VQL**, similar to SQL, allows collection tasks to be quickly programmed, automated and shared, so that **turn-around** from IOC to full hunt can be a few minutes. E.g. VQL to collect files in users' temp directory which have been created within the last week.

Performance, Deployment, Integration and Management

Veramine sensors on average take less than **1% CPU** and **20 MB RAM**, network traffic is less than **30 MB/day/host**, and can be further tuned using collection policies. Easy **deployment** to the whole network such as using AD, SCCM or psexec.

Integration with SIEM, VDI, LDAP, AD, 2-fact Authen, APIs. Sensor Emergency & Autoupdate. Server: Multisite and audited.

Contact: Lan Nguyen, PhD. Co-founder & EVP. Email: lan@veramine.com