Asgard: An Adaptive Self-guarded Honeypot

Sereysethy Touch   Jean-Noël Colin

InfoSec Group, Nadi Research Institute
University of Namur, 5000 Namur, Belgium.

CyberExcellence, March 17, 2022

UNIVERSITÉ DE NAMUR

Background

Definition

A honeypot is a computer tool or resource whose value lies in being probed, attacked, and compromised (Spitzner, 2003)

- It is used for two main purposes: (1) protect the production system, and (2) collect a new attack data.
- Classification of honeypots by their degrees of interaction:
  - Low-Interaction Honeypot (LiHP): an emulator, low risk, low quality data
  - Medium-Interaction Honeypot (MiHP): an emulator, low risk, medium quality data
  - High-Interaction Honeypot (HiHP): a real system, high risk, high quality data

Related Works

- Adaptive (smart) honeypot: uses machine learning techniques to learn to change its behaviour to engage with the attacker.
  - Heliza: HiHP, Modified Linux Kernel (Wagener et al, 2011)
    - allow,
    - block,
    - substitute, and
    - insult.
  - RASSH (Pauna et al, 2014), QRASSH (Pauna et al, 2018): MiHP, using Kippo/Cowrie – a Linux shell emulator
  - IoTCandyJar (Luo et al, 2017) and Dowling's systems (Dowling, 2018): LiHP, they focus on IoT devices

Limitations of the existing systems

What are the limitations of these systems?

- The system like Heliza can be compromised.
- The easy fingerprinting of low- and medium-interaction honeypots due to their limited functionalities.

Contribution

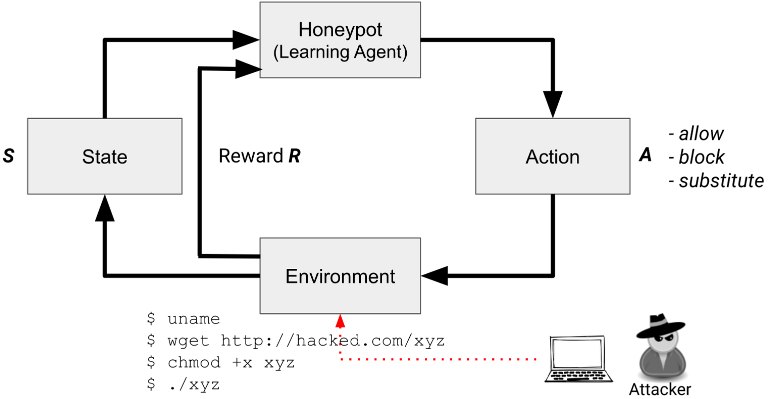
- A new adaptive self-guarded honeypot using the SSH protocol, that leverages reinforcement learning algorithms (RL) to achieve these two objectives:
  1. Interact with the attackers to collect their tools.
  2. Defend itself from being deeply compromised.
- A prototype implementation of the proposed approach by using a simulated botnet attack with real attack data (Touch and Colin, 2021)
- A comparison of our system with the conventional honeypots (Touch and Colin, 2022).

Problem formulation

What is our honeypot?

- A vulnerable Linux system that allows attackers to access it through SSH protocol.
- The attacker interacts with our system by using Linux commands.
- The honeypot has two objectives:
  1. Capture the attackers' tools, and
  2. Guard against a deep system compromise.
- Asgard: An Adaptive Self-guarded Honeypot.

Honeypot as a RL agent



Example of an attack sequence

#	Command	State	Action	Reward
1	`cd /tmp`	*L*	*allow*	0
2	`rm -rf x86*`	*L*	*substitute*	0
3	`wget 107.189.xx.yy/x86_64`	*D*	*allow*	1
4	`chmod 777 *`	*L*	*block*	0
5	`./x86_64 fw.x86`	*C*	*allow*	−1
6	`...`	...	...	...

That was during the learning phase where actions were more randomly chosen.

Example of an attack sequence

And after the learning phase:

#	Command	State	Action	Reward
1	`cd /tmp`	*L*	*allow*	0
2	`rm -rf x86*`	*L*	*substitute*	0
3	`wget 107.189.xx.yy/x86_64`	*D*	*allow*	1
4	`chmod 777 *`	*L*	*block*	0
5	`./x86_64 fw.x86`	*C*	*block*	0
6	`...`	...	...	...

Formal Model Representation

- **Environment state**: a set of Linux command names represented by $S = L \cup D \cup C \cup U$
- **Action**: $A = \{allow, block, substitute\}$
- **New reward function at time-step** t:

$$r_a(s_t, a_t) = \begin{cases} 1 & \text{if } s_t \in D \quad \text{and} \quad a_t \in \{allow\} \\ -1 & \text{if } s_t \in C \quad \text{and} \quad a_t \in \{allow\} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

- **Learning algorithm**: Q-Learning algorithm (Waltkins 1992, Sutton 2018), $q(s, a) \mapsto R$

$$q(s, a) = q(s, a) + \alpha \left[r + \gamma \max_{a'} q(s', a') - q(s, a) \right] \tag{2}$$

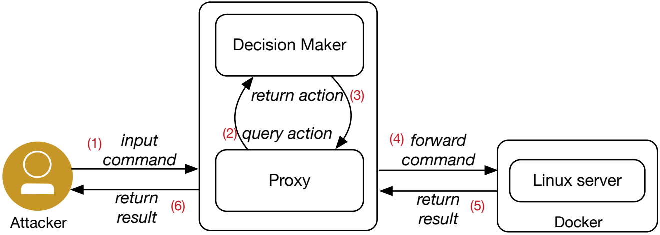
- **Random learning policy**: ϵ-greedy

Midgard: A variant of Asgard

- Midgard shares the same objective as Asgard's.
- The objective is to test a different reward function which will guide the agent toward a different behaviour.
- Its reward function only depends on the state.

$$r_m(s_t, a_t) = \begin{cases} 1 & \text{if } s_t \in D \\ -1 & \text{if } s_t \in C \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

A Proxy-based Architecture



The advantages of this architecture:

- Independence between the honeypot and the target system used as a honeypot.
- No modification of a real system.
- No longer limited to a system emulator.
- Less problem of a system compromission.

Experimental Setup

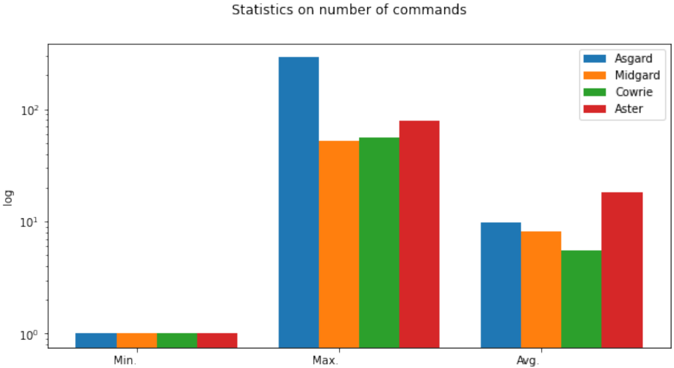
- We deployed 4 honeypots as Docker containers on 4 virtual machines on the same network:
  - Asgard
  - Midgard
  - Aster: a high-interaction honeypot
  - Cowrie: a medium-interaction honeypot
- Host system: Debian 10 buster
- Deployment period: from 11/2021 until early 03/2022 (100 days)

UNIVERSITÉ DE NAMUR

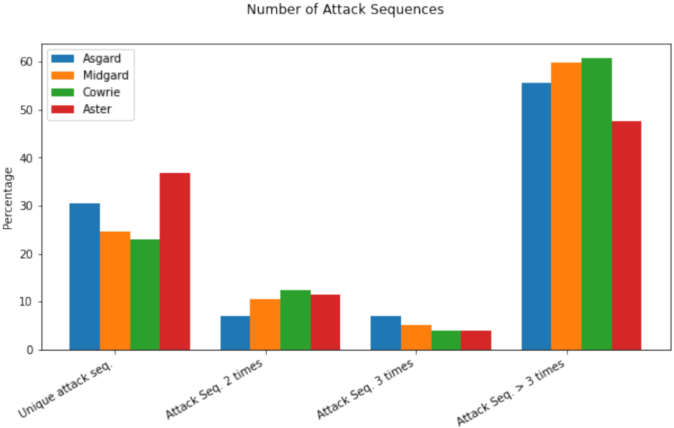
Evaluation criteria

- The number of attack episodes, the number of commands (min., max. and avg.), and the number of attack sequences
- The number and type of collected files
- The attacker's behaviour
- The number of incidents
- The number of human attackers
- The q-values, which show how each adaptive honeypot learns its objectives

Experimental Result 1



Experimental Result 2



The *q*-values of some commands of Asgard

Command	*allow*	*block*	*substitute*
`tar`	**0.3927**	0.1102	0.0983
`sudo`	0.0578	**0.0765**	0.0554
`chmod`	**0.7317**	0.2349	0.2655
`uname`	0.1026	**0.1716**	0.0943
`unknown`	0.0447	**0.0454**	0.0452
`custom`	-0.4058	0.0086	**0.0086**
`ps`	**0.1645**	0.0703	0.0214
`wget`	**1.9696**	0.4153	0.3959
`bash`	0.0134	0.0135	**0.0135**
`sh`	0.1635	0.2392	**1.1545**

The *q*-values of some commands of Midgard

Command	*allow*	*block*	*substitute*
`tar`	**-0.2586**	-0.2589	-0.2613
`sudo`	-0.1699	-0.1842	**0.0957**
`chmod`	-0.4297	-0.4491	**0.3255**
`uname`	**0.9130**	0.1210	0.1461
`unknown`	-0.1308	-0.0793	**0.1480**
`custom`	**-0.9715**	-1.0743	-1.0671
`ps`	**0.2964**	0.0391	0.0535
`wget`	1.0265	**1.3700**	1.0476
`bash`	0.0447	**0.0449**	0.0445
`sh`	0.1075	0.1350	**0.9158**

Lessons learned

- High risk, high return for HiHP *vs.* Low risk, medium return for Asgard.
- A real system *vs.* an emulator (Cowrie), quality over quantity.
- An emulator such as Cowrie still plays an important role.
- Attack trend: monetization *vs.* infection.

Result Discussions

The result of the learned q-values match the two objectives:

1. Allowing the download commands will result in getting attacker's tools.
2. Blocking or substituting the custom commands will protect the honeypot.

Limitations

- Always blocking the custom commands does not allow us to observe the consequences of some attacks.
- Some random actions can be used to fingerprint our system.
- The attacker can still deceive the honeypot by hiding commands in a file.
- The experiment was conducted one time.

Conclusion

- A new adaptive honeypot that can achieve a trade-off between two objectives, thanks to the new reward function.
- A prototype implementation based on a proxy that can solve some problems of the existing honeypots:
  - The modification of a real system,
  - The usage of an emulator,
  - The high risk of security of using a real system.
- The experimental result shows the effectiveness of this system compared with the conventional honeypots.

Future works

- Consider a complex environment state: command, its argument and the honeypot properties (CPU, memory, network bandwidths, . . .)
- Consider an indice of compromise from external IDS to create a richer state.
- Consider a more dynamic reward to allow the execution of a custom program for a certain level of risk.
- Define the command risk levels.

References I

- Touch, Sereysethy and Jean-Noël Colin (2021). "Asguard: Adaptive Self-guarded Honeypot". In: *17th International Conference on Web Information Systems and Technologies-Volume 1: DMMLACS,* SciTePress, pp. 565–574.
- — (2022). "A Comparison of an Adaptive Self-Guarded Honeypot with Conventional Honeypots". In: *Applied Sciences* 12.10, p. 5224.

Paper 1:

Paper 2:

UNIVERSITÉ DE NAMUR