SuperviZ

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Toward Fast and Autonomous Security Reaction Mechanisms: Application to Malware Propagation

SOTERN - IRISA, IMT ATLANTIQUE

- Do Duc Anh NGUYEN (Presenter) - do-duc-anh.nguyen@imt-atlantique.fr
- Pierre ALAIN - pierre.alain@irisa.fr
- Fabien AUTREL - fabien.autrel@imt-atlantique.fr
- Ahmed BOUABDALLAH - ahmed.bouabdallah@imt-atlantique.fr
- Guillaume DOYEN - guillaume.doyen@imt-atlantique.fr
- Jérôme FRANÇOIS - jerome.francois@uni.lu

SuperViz meeting, Paris, March 11, 2025

Outline

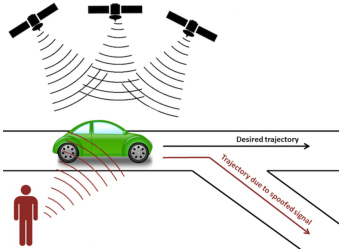
Outline

Why Security Automation?

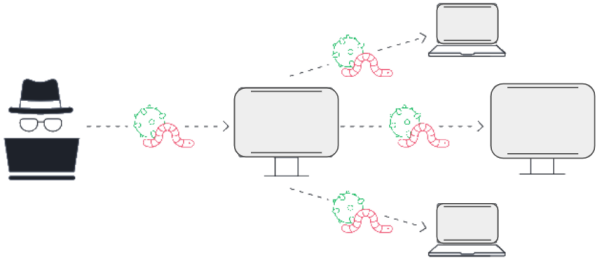
- Escalating Cyber Threats
- Faster Incident Response
- Reduced Human Errors

Research question

Are current security reaction mechanisms effective in mitigating rapid attack events?



GPS spoofing

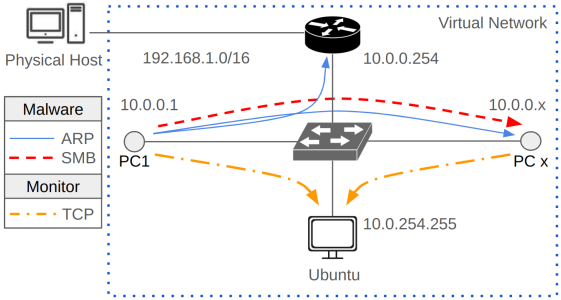


Malware propagation → Selected usecase

Outline

Previous Work [1]

The propagation behavior of WannaCry and NotPetya has been empirically studied



Contributions

- Measurement of propagation speed
- Discussion on their propagation strategies

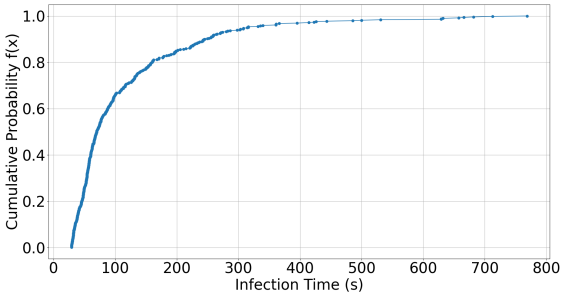
→ Provide meaningful insights into malware propagation on a local network and on the Internet

Local network with 50 Windows hosts

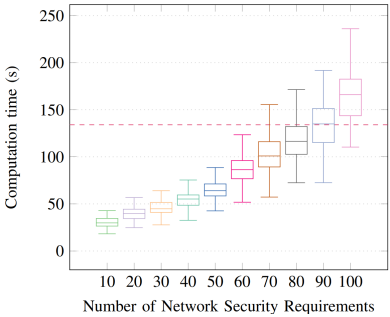
[1] Do Duc Anh Nguyen et al. ”How Fast does Malware Leveraging EternalBlue Propagate? The case of WannaCry and NotPetya“. In: SecSoft Workshop. 2024

Problematic

Are current security reaction mechanisms effective in mitigating malware propagation?



Nearly 60% infections $<$ 100s (WannaCry)



VEREFOO [2] process 70 user requirements in 100s

Centralized approaches can be slow and unscalable in preventing malware propagation

[2] Daniele Bringhenti et al. "Automated Firewall Configuration in Virtual Networks". In: IEEE Transactions on Dependable and Secure Computing 20.2 (2023)

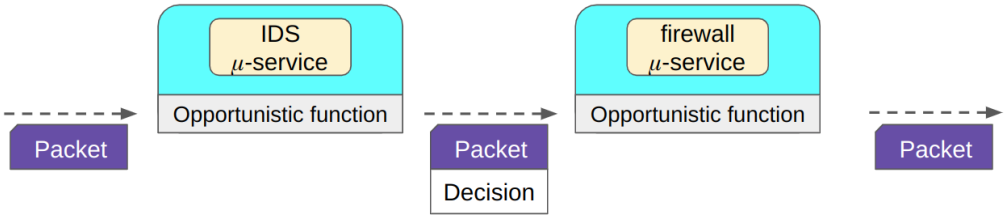
Proposal

Flexibility and scalability in security function deployment

Microservices [3]

Microservices are based on the concept of breaking complex applications into multiple small services that handle specific functions and can be modified without affecting the others.

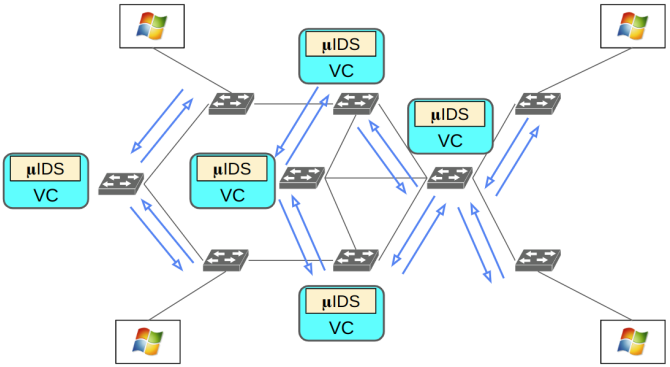
Fast reaction decision transmission: Decentralized and opportunistic synchronization



$\Rightarrow$ Not the optimal solution, but can stop the attack immediately

Research Questions

How can we optimize the placement of IDS microservices? **A Vertex Cover (VC) approach**

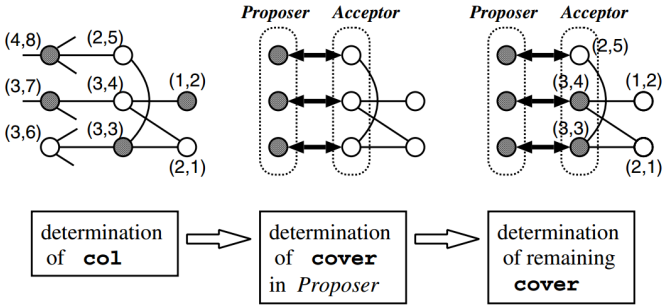


The distributed VC algorithm [4] is selected

- Making local decisions with local information
- Near-optimal solutions

[4] Jun Kiniwa "Approximation of self-stabilizing vertex cover less than 2". Symposium on Self-Stabilizing Systems (2005)

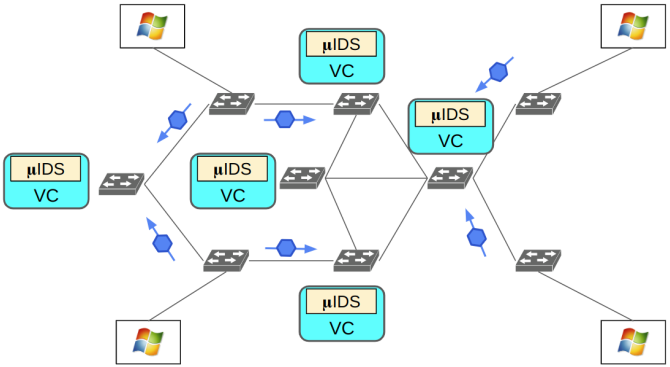
The Self-Stabilizing VC Algorithm



1. If I was not matched $\Rightarrow col_i := d_i$
2. If I was matched $\Rightarrow col_i := max(d_i, d_k)$
3. If a higher-color node j point to me $\Rightarrow i \rightarrow j$; $col_i := d_j$; $cover_i := false$
4. If I point to a higher-color node but he does not point to me $\Rightarrow i \rightarrow null$; $col_i := d_i$; $cover_i := false$
5. If no one point to me and I have a smaller-color node $j \Rightarrow i \rightarrow j$; $cover_i := true$
6. If I am not the smallest-degree node $\Rightarrow cover_i := True$, otherwise, $cover\_i := False$

Research Questions

How can we block traffic as close to the attack source as possible? **IP traceback**



Adapt the [5] solution to allow μIDS traceback to where the attack source is

- Non-VC nodes embed its ID into packets for traceback

[5] Runhu Wang et al. "In-band network telemetry based fine-grained traceability against IP address spooling attack". ACM (2021)

Outline

Demonstration: Opportunistic Reaction - The Case of WannaCry

To what extent microservices augmented with opportunistic synchronization can mitigate the WannaCry propagation in a basic but realistic environment

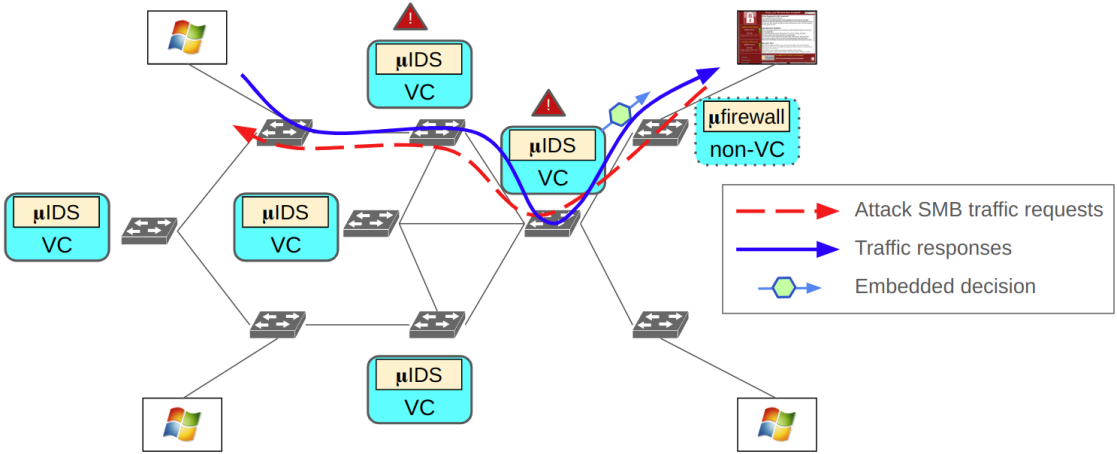
Setup

- 4 Windows hosts, 1 of which contains the WannaCry executable
- IDS and firewall microservices: MirageOS unikernels vs ~~Unikraft unikernels~~
- Switch emulation in GNS3: Ubuntu VMs

Scenario

- The distributed VC algorithm
  - ▷ Each node collects its local information by broadcasting
  - ▷ They decide to be in VC set (deploy an IDS) or not
- The opportunistic approach: The option field in the IP layer is used
  - ▷ Non-VC nodes: their ID, interface number, registration (i.e., 1)
  - ▷ VC nodes: node ID, interface number, DROP decision (i.e., 0)

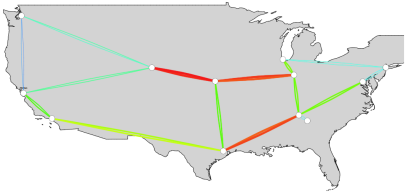
Demonstration: Opportunistic Reaction - The case of WannaCry

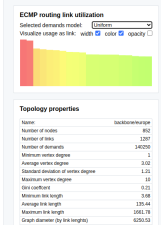


Outline

Ongoing Work

Putting our approach into more realistic scenarios
Layer-3 topologies shared by topohub [6]





Information on the topohub

- Multiple real topologies of different sizes
- Number of network nodes, locations
- Traffic volume, link utilization

[6] Piotr Jurkiewicz et al. "TopoHub: A repository of reference Gabriel graph and real-world topologies for networking research". SoftwareX. 2023

Ongoing Work

Generate legitimate traffic using traffic models

- Source models
  - Poisson model: Input mean rate
  - On/Off model: Input pack rate, probabilities of on/off process

  $\rightarrow$ Distribution of packet arrivals
- Source-Destination models
  - Gravity model: Input volumn of traffic for each host
  - **Indenpendent-Connection (IC) model** [7]: Input traffic volume, popularity, ratio of forward and reverse traffic for each host

  $\rightarrow$ Results in a Traffic Matrix (TM), which represents the volume of traffic from a host to other hosts

  $\Rightarrow$ Can take the TM directly into traffic generators

[7] Vijayi Erramill et al. "An independent-connection model for traffic matrices". ACM. 2006

Outline

Future Work

Propose an empirical performance evaluation to the Journal of Network and Systems Management (JNSM)
Research questions

- How can we block attacks when requests and responses take different paths?
- How can we block attacks when no IDS sits between attacker and victim?
- How can we delegate knowledge of a removed IDS to other IDSs?
- How to secure the opportunistic communication?

Ending

Thank you for listening
Source code:
`https://gitlab.imt-atlantique.fr/d22nguye/gns3_unikernel_testbed`