Effectiveness of Proactive Reset for Mitigating Impact of Stealthy Attacks on Networks of Autonomous Systems

Brian Thompson
U.S. Army Research Lab
Adelphi, MD 20783
Email: bthompso8784@gmail.com

James Morris-King
U.S. Army Research Lab
Adelphi, MD 20783
Email: james.r.morris-king.ctr@mail.mil

Hasan Cam
U.S. Army Research Lab
Adelphi, MD 20783
Email: hasan.cam.civ@mail.mil

***Abstract*—Recent examples have shown that sophisticated cyber attackers are capable of infiltrating the cyber defenses of major organizations and spreading stealthily through a network, potentially doing significant damage before exploited vulnerabilities can be identified or patches developed. Autonomous systems are particularly vulnerable because they are further removed from human intervention. One emerging technology designed to address this problem is proactive reset, where systems automatically undergo a reset operation that restores them to a known malware-free state, regardless of whether or not they were already infected. More frequent resets result in higher security, but may also reduce functionality of the network. In this work, we consider the effectiveness of three proactive reset policies for mitigating the spread of stealthy malware through a network of autonomous systems. We perform experiments using agent-based simulation and find that a proactive policy that uses risk-flow analysis to determine when systems should be reset achieves performance comparable to that of a perfect detector.**

I. Introduction

A. Motivation

Networks of autonomous systems—such as robotic factory workers, security robots, and unmanned aerial vehicles (UAVs, commonly known as drones)—are increasingly being used to perform tedious, demanding, or dangerous tasks with limited or no human intervention. Their autonomy and interconnectedness, however, increases their susceptibility to cyber attack as well as the magnitude of damage an attack could cause.

In 2011, a computer virus was detected that logged keystrokes from U.S. Air Force pilots remotely controlling UAVs on missions in Afghanistan and other foreign countries [1]. In 2012, a Drone Games participant designed a drone that wirelessly infects other drones with malware that hijacks their control systems and uses them to spread the malware to additional drones [2]. In 2013, a drone called SkyJack was designed that autonomously identifies and breaks into nearby drones wirelessly [3]. In 2015, a security researcher developed a backdoor for drones called Maldrone, which once installed, takes control over navigation and other systems [4]. With the increasing prevalence of autonomous systems in military, commercial, and recreational use, it is essential to understand the dangers, assess risk, and develop policies to effectively control the spread of malware through such networks.

Many existing cybersecurity solutions are reactive, such as signature-based anti-virus software, intrusion detection systems, or the patching of discovered vulnerabilities, and do not prescribe any defensive action if nothing has been detected. However, some attacks may go through an initial spreading phase during which changes in the behavior of infected devices are minimal, making the presence of the malware difficult to detect. Furthermore, new attacks are developed continually, and vulnerabilities often go undiscovered until successful attacks have already been executed, at which point much of the damage has already been done.

As a defensive maneuver, systems can undergo a resetting process designed to remove potential malware. A variety of existing and emerging technologies deliver such capabilities, depending on the context and type of device [5], [6]. In some contexts, resetting could be a simple software restart; in others, it could entail reimaging the disk or switching to a new virtual machine created from a clean disk image. However it is implemented, the essential requirement is that the reset operation returns the system to a malware-free state.

When a system is reset, it becomes unavailable for a period of time. Resetting non-infected systems is therefore undesirable because it diminishes functionality without improving the security of the network. On the other hand, the longer an infected system remains in operation, the greater the chance that the malware will spread to other systems on the network. Therefore, it is important to determine appropriate conditions under which systems should be reset.

In this work, we present a proactive approach to cybersecurity that dynamically schedules networked systems to be reset. Our approach is proactive in the sense that it prescribes taking defensive actions even when no malicious activity is suspected. We model the spread of self-propagating malware through a network of autonomous systems, with the ability to take control over infected systems, and an automated defense mechanism that resets systems according to a reset policy. We propose several policies designed to mitigate the spread of stealthy malware while obeying a constraint on the required minimum number of operational systems at any time. We compare the policies' effectiveness in purging propagating malware from a network through experiments using agent-based simulation and find that a proactive policy that uses

risk-flow analysis to determine when systems should be reset achieves performance comparable to that of a perfect detector.

B. Related Work

1) Control of malware spread: Okhravi and Nicol evaluate the tradeoff between the time spent on pre-deployment testing and the timely deployment of patches for software vulnerabilities [7]. Khouzani et al. explore how to allocate resources to prevent the spread of an aggressive malware infection controlled by an attacker seeking to do maximum damage to a mobile wireless network [8]. Eshghi et al. propose patching strategies for countering propagating malware in both a replicative (patches can be transmitted by other patched devices) and a non-replicative (patches are only disseminated by designated sources) context [9].

These approaches rely on the existence of patches for known vulnerabilities, whereas our proactive approach is also effective in combating stealthy attacks that may be exploiting unknown vulnerabilities.

2) Dynamic scheduling in networks: There is a large body of literature on scheduling policies for distributed network settings. Chase et al. propose an architecture for dynamically resizing the set of active servers in a computing cluster [10]. Ranjan et al. propose algorithms for dynamic resource allocation by migrating data between servers in distributed databases [11]. Hong et al. study dynamic server provisioning for low-cost cloud computing [12].

These policies are typically designed for a non-adversarial context, whereas our policies are designed to defend against a malicious attacker spreading optimally through a network.

3) Proactive cyber defense: Evans et al. evaluate the effectiveness of moving target defense against a variety of real-world attacks [13]. Colbaugh and Glass take a game-theoretic approach and propose moving target defense strategies against an adaptive attacker [14]. Ben-Asher et al. study the effectiveness of migration-based moving target defense against attackers with a range of skills and resources [15]. Shan et al. propose proactive restart of smartphone apps to reduce side channel time series predictability [16].

These works all consider proactive defense mechanisms for a single system rather than a coordinated effort over networked devices, and are therefore not sensitive to the needs of the network as a whole. To our knowledge, we are the first to propose a viable approach to security in cyber networks that uses proactive reset to achieve cybersecurity goals while obeying resource constraints on a network-wide scale.

C. Contributions and Outline

The main contributions of this work are:

- A model for analyzing the security of networks of autonomous systems that incorporates elements of network communication, malware propagation, and a defensive reset operation
- A proactive reset policy based on the concept of risk flow that achieves performance comparable to that of a perfect detector by leveraging the autonomous systems' communication histories
- Evaluation of the proposed reset policies and comparison to baselines using agent-based simulation

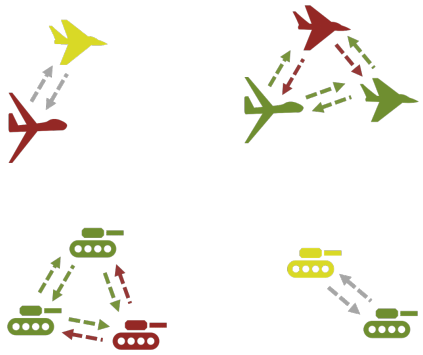


Fig. 1: A sample network illustrating our model. Green nodes are clean. Red nodes are infected. Yellow nodes are resetting. A stealthy cyber attack spreads from infected nodes to clean nodes over existing communication.

In Section II we describe our network model, define the problem of interest, and propose three classes of proactive reset policies. In Section III we compare the performance of our proposed policies to that of a hypothetical perfect detector using agent-based simulation. We conclude with discussion and ideas for future work in Section IV.

II. Methods

A. Model and Definitions

We model a network of interacting *nodes* corresponding to autonomous systems cooperating in order to carry out a common task. When operational, pairs of nodes periodically communicate with one another, providing a channel for data transfer between them. We assume that all nodes have the same capabilities, and that the network can tolerate the inoperability of some of the nodes.

We consider an attacker trying to stealthily establish and maintain a presence on nodes in the network, which could then be used at an opportune time for malicious purposes such as preventing task completion or causing physical damage. The attacker begins with a presence on an initial set of *seed* nodes and spreads stealthily to other nodes via self-propagating malware, which transmits itself over existing communication channels. If the attacker has established a presence on a node, we say that the node is *infected*, otherwise we say it is *clean*. By only spreading across existing communication channels, the attacker hides evidence of its progress. Our network model is illustrated in Figure 1.

We consider an automated defense mechanism that can choose to perform a *reset* operation on any node, which restores it to a known malware-free state but renders it inoperable in the meantime. The implementation is domain-specific and outside the scope of this paper. It should be

understood that our approach will only be effective against malware that can be removed from a device by performing the designated reset operation.

Let r denote the *reset time*, the time required to reset a node and return it to operational status. When the reset is complete, we say that the node is *activated*.

Let θ denote the *operational threshold*, the minimum fraction of nodes required to be operational in order to sustain the desired level of network functionality.

A *reset policy* dictates which–if any–nodes should be reset at any given time.

Problem Statement: Given parameters r and θ corresponding to the reset time and operational threshold, respectively, design a reset policy to minimize the number of infected nodes.

B. Reset Policies

We consider three classes of proactive reset policies: Random Reset, Communication-Based, and Risk-flow.

Random Reset policy: Under this policy, nodes are reset at random until the operational threshold has been reached. The intuition is that eventually all nodes will be reset, and the likelihood that any one node will not be reset for a long time is exponentially small.

Communication-based policy: Nodes are prioritized for resetting by the amount of communication they have had since their last reset. The intuition is that the more incoming communication the user has received, the greater the likelihood that the node has been compromised.

Risk-flow policy: Nodes are prioritized by a risk score that is maintained for each node. The risk score serves as a proxy for the likelihood that the node is compromised. When a node is reset, it gets activated with an initial value of 1. When two nodes communicate, each node updates its risk score to be the sum of their old risk scores, reflecting the fact that it will now be compromised if either node was compromised previously.

In the following section, we turn to agent-based simulation to compare the effectiveness of our proposed proactive reset policies for limiting the spread of propagating malware.

III. Evaluation

A. Simulation Model

We develop and implement a discrete-time agent-based simulation model in Java based on the network model presented in Section II-A. We perform experiments on a network of 100 nodes and let each simulation run for 10,000 time steps, corresponding to 1,000 minutes (16 hours, 40 minutes).

In practice, the network parameters r and θ, corresponding to the reset time and operational threshold, respectively, should either be known or estimated based on past observations or domain knowledge, and may be system- and context-dependent. In our experiments, the reset time is set to 1 minute and the operational threshold varies from 50% to 100%.

To simulate communication, we consider the worst case scenario in which any node can communicate with any other node, which leads to the greatest degree of uncertainty about the spread of the malware. In our experiments, each node communicates with a random node in the network on average once every 1 minute.

We consider two initial configurations: (1) The attacker begins with one seed node, representing a single point of entry into the network. (2) The attacker begins with all nodes being infected, representing a scenario in which the attacker has already established a large presence on the network at the time the reset policy goes into effect.

We now use our agent-based simulation model to evaluate the effectiveness of our proposed reset policies.

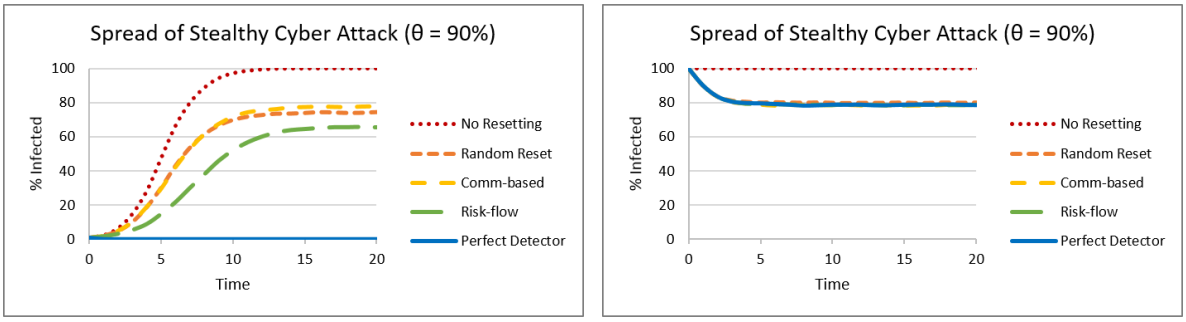
B. Results

We compare the performance of the Random Reset, Communication-based, and Risk-flow policies to two baseline policies:

- No Resetting: Never reset any node
- Perfect Detector: Only resets infected nodes

Figures 2, 3, and 4 show the progress of the attacker over time when the network has a required operational threshold of 90%, 80%, and 70%, respectively, averaged over 100 trials. We see that in all cases, after a period of time the system seems to reach an equilibrium. As expected, under the No Resetting policy, the attacker quickly spreads to the entire network. The Random Reset policy limits the fraction of infected nodes, but is not able to purge the attack from the network, even at a 70% operational threshold. When $\theta = 80\%$, the Communication-based policy keeps the fraction of infected nodes below 50%, but when only 70% of the nodes are required to be operational, the Communication-based policy is able to completely eliminate the infection. For $\theta \leq 80\%$, the Perfect Detector successfully eliminates the attacker's presence, although when the initial configuration is full infection of the network, it takes time — about 5 minutes when only 70% of the nodes are required to be operational, and over an hour and a half when the operational threshold is 80%. Strikingly, the Risk-flow policy achieves nearly the same performance as the Perfect Detector using only information about the communication histories of the nodes, eliminating the malware in about two hours when $\theta = 80\%$ and about 10 minutes when $\theta = 70\%$ when starting from full infection.

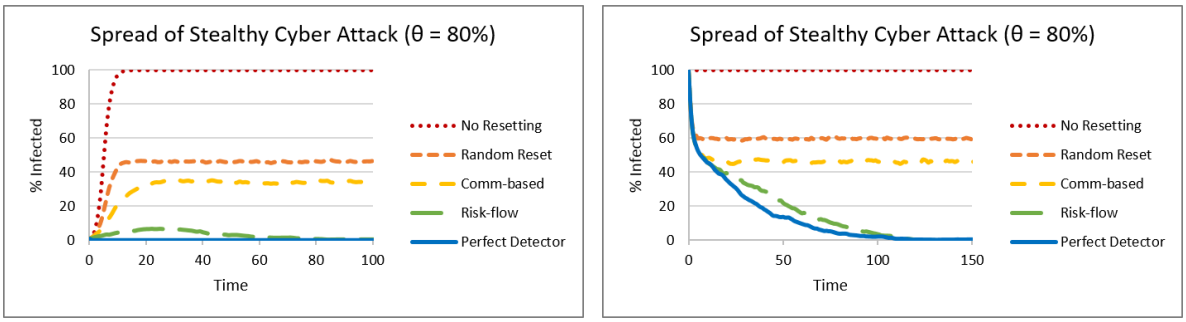
Figure 5 shows the fraction of infected nodes at the end of the simulation as the operational threshold varies, averaged over 100 trials. The results are different for the two different initial configurations (point infection and full infection) because with a point infection, occasionally the defender will get lucky and reset the only infected nodes while the attack is still in its initial stages, resulting in 0% infected nodes at the end of the simulation and thus lowering the average over all trials. The No Resetting policy never eliminates malware from any nodes, so is a constant 100% throughout. For $\theta > 60\%$, the Random Reset policy limits the fraction of infected nodes roughly proportionally to θ, but for $\theta \leq 60\%$, it is eventually



(a) 1 seed node (point infection)

(b) 100 seed nodes (full infection)

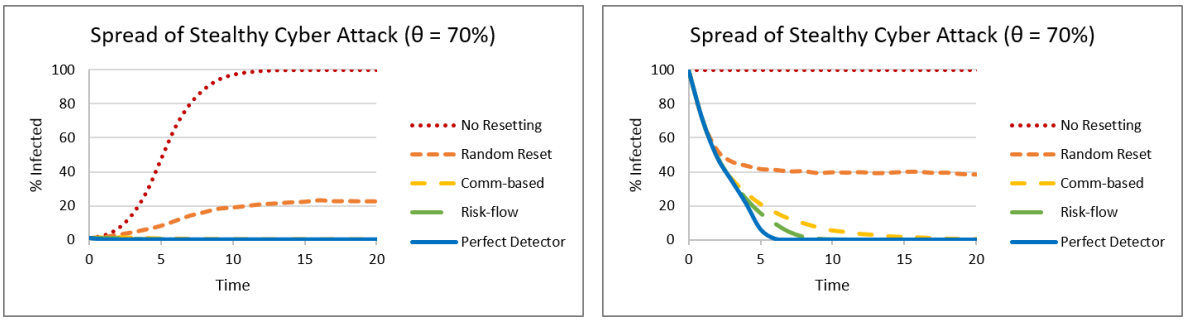
Fig. 2: Progress of the attacker over time, in terms of the fraction of infected nodes, with $\theta = 90\%$.



(a) 1 seed node (point infection)

(b) 100 seed nodes (full infection)

Fig. 3: Progress of the attacker over time, in terms of the fraction of infected nodes, with $\theta = 80\%$.



(a) 1 seed node (point infection)

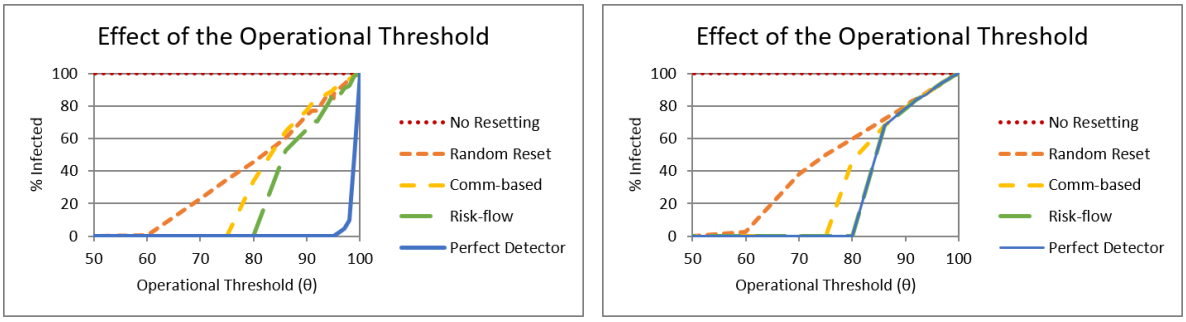
(b) 100 seed nodes (full infection)

Fig. 4: Progress of the attacker over time, in terms of the fraction of infected nodes, with $\theta = 70\%$.

able to completely purge the attack from the network. The Communication-based policy behaves similarly but reduces the fraction of infected nodes at roughly twice the rate of the Random Reset policy, successfully eliminating attacks for $\theta \leq 75\%$. Under an attack with a point infection, the Perfect Detector can easily rid the network of the attacker's presence as long as the operational threshold is less than 95%. Starting from full infection, however, even the Perfect Detector cannot fully eliminate an attack for $\theta > 80\%$. The Risk-flow policy is not able to take advantage of point infections because initially it can not identify the nodes that are most likely to be infected, but starting from full infection, it achieves the same performance at the Perfect Detector, successfully purging malware from the network for $\theta \leq 80\%$.

IV. Conclusions

In this work we introduced a proactive approach to cybersecurity in networks of autonomous systems. We presented a model that captures communication between devices, stealthy propagation of malware, and a defensive reset operation that removes malware from a device at the cost of



(a) 1 seed node (point infection)

(b) 100 seed nodes (full infection)

Fig. 5: Fraction of infected nodes at the end of the simulation as θ varies. Results not shown for $\theta < 50\%$ because they are the same as for $\theta = 50\%$.

reduced availability to perform normal network functions. We proposed three proactive reset policies—Random Reset, Communication-based, and Risk-flow—to limit the number of infected devices in the presence of a spreading cyber attack. We developed an agent-based simulation model to evaluate the effectiveness of our proposed reset policies.

Experimental results showed that even a naive proactive policy provides significant benefits, and that a proactive policy that leverages nodes' communication histories to estimate each node's risk of infection can achieve nearly identical performance to that of a perfect detector. However, all of these policies—even a perfect detector—have their limits; the defender must be willing to tolerate that a fraction of the devices will be temporarily inoperable, otherwise the policies will be ineffective. For sufficiently low operational requirements, a good detector or a proactive policy will eventually be able to completely purge propagating malware from a network.

Directions for future work include considering arbitrary network structures instead of assuming all-pairs reachability, incorporating a mobility model, and modeling heterogeneous devices with different risks, functionality, and requirements. It would also be of potential interest to develop a hybrid policy that combines elements of our proactive approach with detection-based methods in order to optimize performance for known malware while still providing a measure of security against stealthy attacks.

References

[1] N. Shachtman, "Computer virus hits u.s. drone fleet," https://www.wired.com/2011/10/virus-hits-drone-fleet/, 2011.

[2] A. Tarantola, "This virus-copter is a digital typhoid mary," http://gizmodo.com/5967209/this-virus-copter-is-a-digital-typhoid-mary, 2012.

[3] J. Condliffe, "Skyjack lets you hunt down and hack other drones from the air," http://gizmodo.com/skyjack-lets-you-hunt-down-and-hack-other-drones-from-t-1476286537, 2013.

[4] A. C. Estes, "New malware can bring down drones mid-flight," http://gizmodo.com/new-malware-can-bring-down-drones-mid-flight-1682100201, 2015.

[5] A. M. Dunn, M. Z. Lee, S. Jana, S. Kim, M. Silberstein, Y. Xu, V. Shmatikov, and E. Witchel, "Eternal sunshine of the spotless machine: Protecting privacy with ephemeral channels," in *Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 61–75. [Online]. Available: http://dl.acm.org/citation.cfm?id=2387880.2387887

[6] K. Sun, J. Wang, F. Zhang, and A. Stavrou, "Secureswitch: Bios-assisted isolation and switch between trusted and untrusted commodity oses," in *NDSS*, 2012.

[7] H. Okhravi and D. Nicol, "Evaluation of patch management strategies," *International Journal of Computational Intelligence: Theory and Practice*, vol. 3, no. 2, pp. 109–117, 2008.

[8] M. Khouzani, S. Sarkar, and E. Altman, "Maximum damage malware attack in mobile wireless networks," *Networking, IEEE/ACM Transactions on*, vol. 20, no. 5, pp. 1347–1360, 2012.

[9] S. Eshghi, M. Khouzani, S. Sarkar, and S. Venkatesh, "Optimal patching in clustered malware epidemics," *Networking, IEEE/ACM Transactions on*, vol. 24, no. 1, pp. 283–298, Feb 2016.

[10] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle, "Managing energy and server resources in hosting centers," in *Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles*, ser. SOSP '01. New York, NY, USA: ACM, 2001, pp. 103–116. [Online]. Available: http://doi.acm.org/10.1145/502034.502045

[11] H. F. S. Ranjan, J. Rolia and E. Knightly, "Qos-driven server migration for internet data centers," in *Proceedings of the Tenth IEEE International Workshop on Quality of Service*, 2002.

[12] Y.-J. Hong, J. Xue, and M. Thottethodi, "Dynamic server provisioning to minimize cost in an iaas cloud," in *Proceedings of the ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS '11. New York, NY, USA: ACM, 2011, pp. 147–148. [Online]. Available: http://doi.acm.org/10.1145/1993744.1993799

[13] D. Evans, A. Nguyen-Tuong, and J. Knight, "Effectiveness of moving target defenses," in *Moving Target Defense*. Springer, 2011, pp. 29–48.

[14] R. Colbaugh and K. Glass, "Predictability-oriented defense against adaptive adversaries," in *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*, Oct 2012, pp. 2721–2727.

[15] N. Ben-Asher, J. Morris-King, B. Thompson, and W. J. Glodek, "Attacker skill, defender strategies, and the effectiveness of migration-based moving target defense in cyber systems," in *Proceedings of the International Conference on Cyber Warfare and Security*. ACPI, 2016.

[16] Z. Shan, I. Neamtiu, Z. Qian, and D. Torrieri, "Proactive restart as cyber maneuver for android," in *Proceedings of the Conference on Military Communications*, 2015.