Chapter 8
Robustness to Malware Spreading

While many networked computing systems are vulnerable to self-propagating malicious software, or malware, large enterprises use automated patching and hardening to make their systems highly immune to malware infections. Still, persistent human attackers compromise enterprise networks utilizing advanced tools, customized malware, and zero-day exploits that anti-malware technology and patching cannot detect and mitigate [75, 76]. The three chapters in Part III study how the diversity and fail fast principles from Chap. 4 can be exploited to achieve anti-fragility to malware spreading in networked systems. The current chapter investigates software diversity's ability to make systems robust to the spreading of infectious malware and argues that diversity increases the time needed to compromise enterprise systems, thus increasing the probability of early detection and mitigation. The two next chapters extend the results in this chapter to achieve anti-fragility to malware spreading.

8.1 Introduction

We view a computing system as a collection of interconnected computing devices and consider the devices at the operating system (OS) and application levels. Compilers with "diversity engines" generate the devices' binary images, producing many different executable images from a much smaller set of OS and application source codes [24]. Many techniques to diversify binary images exist [77]. A transformation can be as simple as adding no-operation instructions (NOPs) to an image. The insertion of NOPs is always possible and allows us to produce infinitely many binary variants. Conceptually, a program's binary images are divided into classes such that all members of the same class share at least one exploitable vulnerability, while members of different classes have no common exploitable vulnerabilities. Assuming that the classes are roughly equally large, the number of classes measures the program's diversity with the convention that a network with only a single type, that is, a software monoculture, has no diversity [50].

K.J. Hole, *Anti-fragile ICT Systems*, Simula SpringerBriefs on Computing 1,
DOI 10.1007/978-3-319-30070-2_8

Using well-established network models from network science [23], we combine software diversity and computer "immunization" to halt multiple simultaneous outbreaks of infectious malware with sparse and inhomogeneous spreading patterns, represented by synthetic and empirical networks. We establish an explanatory epidemiological model of variable diversity, determine a general lower bound on the diversity needed, evaluate the halting technique's performance for worst-case spreading over sparse and dense homogeneous networks, consider diversity's ability to slow down persistent threats, and discuss independent research on software diversity.

This chapter proposes and analyzes a halting technique for malware with known static spreading mechanisms. The technique assumes that a small percentage of the nodes can be immunized, that is, made resistant to the malware. In practice, immunization, or hardening, includes the removal of non-essential software programs, the secure configuration of remaining programs, constant patching, and the use of firewalls and intrusion prevention systems. The author first presented the halting technique in [56]. Chapter 10 generalizes the halting technique to malware with unknown and time-varying spreading mechanisms.

8.2 Explanatory Epidemiological Model

Different malware strains exploit vulnerabilities in OSs and application software to infect computing devices. An exploitable vulnerability is a mistake in the software that enables malware to gain access to a device and its information. Examples of exploitable vulnerabilities are buffer overflows and malformed URLs (see [24, 78, 79] for more information on vulnerabilities). Infectious malware can spread to new vulnerable devices via network shares, removable media, Internet protocol (IP) attacks, email messages, instant messaging, and peer-to-peer networks.

8.2.1 Epidemiological Model

We model the spreading of infectious malware over networked computing devices by a simple graph (no self-loops or parallel edges) with N nodes of L types, $1 \leq L \leq N$, as depicted in Fig. 8.1. There are roughly N/L nodes of each type uniformly distributed over the graph. The node types represent different binary codes at the OS or application level of the computing devices; that is, nodes of the same type share an exploitable vulnerability while nodes of different types have no common exploitable vulnerabilities. The edges represent communications between nodes. A good measure of the model's *diversity* is the number of node types L (see [50] for a thorough discussion of diversity). Two nodes are neighbors if they share an edge. A node's degree k is the number of neighbors and $\langle k \rangle$ denotes the nodes' average degree. A network is *homogeneous* when all nodes have degrees $k \approx \langle k \rangle$ and *inhomogeneous* when a small fraction of nodes, called *hubs*, have $k \gg \langle k \rangle$.

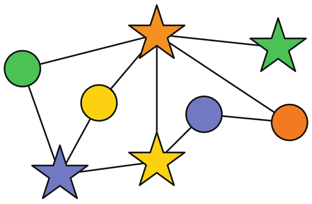

Fig. 8.1 A network with $N = 8$ nodes, $L = 4$ node types of different colors, and average degree $\langle k \rangle = 2.5$. *Stars* represent the infected seeds. There is $S = 1$ seed per node type. Only the *orange* seed will infect a neighbor

The malware's different spreading mechanisms determine the topologies of the spreading networks. Malware utilizing random scanning to target IP addresses spread over nearly fully connected homogeneous networks, while malware utilizing topological scanning travel over inhomogeneous networks [80]. Topological scanning relies on information contained in infected hosts to locate new targets, including routing tables, email addresses, and Uniform Resource Locators (URLs). The resulting virtual spreading networks are different from the physical networks of wired and wireless communication links. We study multiple malware, or *multimalware*, outbreaks because the deployment of several malware types is an obvious strategy to counter software diversity. All malware types are assumed to have the same spreading mechanism.

Our discrete-time model contains L types of infectious malware, that is, one malware type per node type. Each malware type exploits a particular vulnerability to infect a single node type. Initially, S nodes of each type are infected. These $L \cdot S$ nodes are called *seeds* (see Fig. 8.1). The infection probability determines the rate at which a sick node infects a susceptible neighbor of the same type during a time step. To study worst-case spreading, we set the infection probability to one to ensure that all nodes reachable from the seeds are infected. No infected node recovers.

8.2.2 *Non-predictive Model*

It is hard to estimate the actual spreading of malware in a networked computing system because it is influenced by many factors, including router policies, the choice of communication protocols, available bandwidth, traffic loads, firewall rules, anti-malware signature sets, intrusion detection, the level of software patching, and the misconfiguration of system parts. Rather than trying to incorporate all these factors, the epidemiological model displays very fast worst-case spreading where an infectious node always infects all of its neighbors of the same type. While this model cannot predict actual spreading in a network, it can explain the usefulness of software diversity. Because actual malware is likely to spread less, it is reasonable to believe that the model's malware halting translates into malware halting in real systems. This view is supported by independent research discussed later in the chapter.

8.3 Malware-Halting Technique

The following proposed malware-halting technique immunizes hubs if they exist and increases the diversity L to limit the fraction of infected nodes:

1. If the spreading network is inhomogeneous, immunize enough large-degree nodes to create a homogeneous subnet when the immunized nodes and their adjacent edges are removed.
2. Ensure that the node diversity of the homogeneous subnet is large enough to halt multiple simultaneous malware outbreaks.

Table 8.1 outlines how to halt multimalware outbreaks on sparse (small $\langle k \rangle$) or dense (large $\langle k \rangle$) networks with homogeneous or inhomogeneous topologies. Limited *true* diversity (small L) is obtained by deploying instances of different OSs and applications with similar functionality. Michael Franz [24] argues that much greater *artificial* diversity (large L) is available when users download software from application stores utilizing compiler-generated diversity to produce many classes of executable binary images. While true diversity is costly because the installation of different software forces users to learn new functionality, the cost of artificial diversity is reasonable, since the functionality is not changed.

An example illustrates the halting technique on a sparse and inhomogeneous network with hubs. Figure 8.2a shows a synthetic network with 300 nodes. The nodes are circles with areas proportional to their degrees, thus highlighting the hubs. The spreading network is a software monoculture with one node type ($L = 1$) [28, 29]. All nodes are red to illustrate that a single seed ($S = 1$) infects all nodes. Figure 8.2b shows the same network, but now with randomly distributed orange and yellow node types ($L = 2$). Eight white hubs are made immune to two malware types attacking the nodes. There is little malware spreading in this immunized "polyculture." For a particular selection of two seeds of different types, Fig. 8.2b shows that the malware spreading is reduced from 300 nodes to only three red nodes; that is, the halting technique decreases the percentage of infected nodes from 100 to 1 %.

The simple illustrative spreading network in Fig. 8.2 has no loops and the hubs are connected in a small subnetwork. During the following analysis, we consider networks with loops and make no assumptions about how the hubs are connected.

Table 8.1 Malware halting on spreading networks with different topologies

Malware halting on different network topologies		
Sparse and homogeneous	Sparse and inhomogeneous	Dense and homogeneous
Utilize small true or artificial diversity	Use hub immunization and small true or artificial diversity	Deploy large artificial diversity

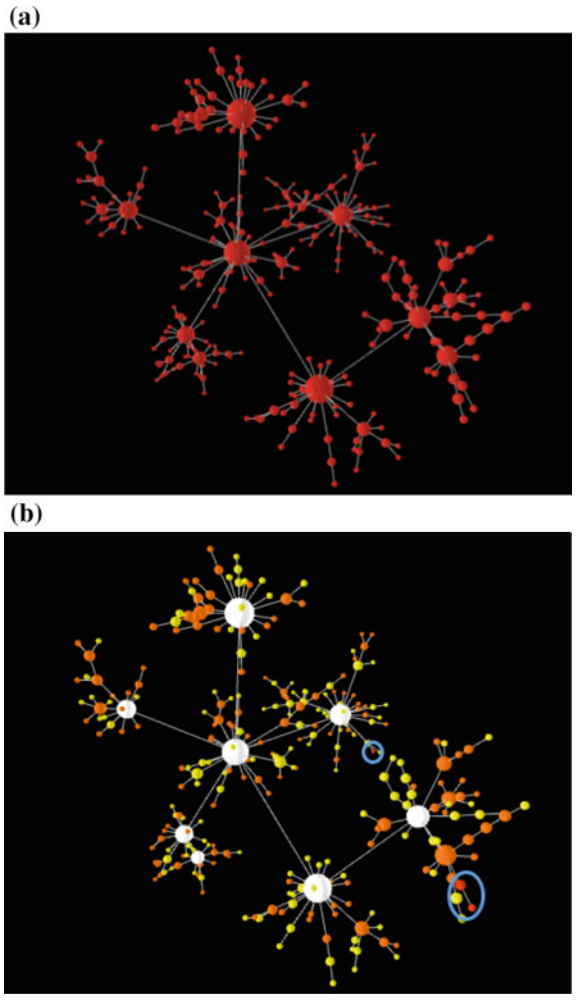

Fig. 8.2 **a** Monoculture with 300 infected nodes whose areas are proportional to their edge degrees. **b** The same network as in (**a**) but with *white* immunized hubs and *orange* and *yellow* node types. Two malware types, each with a single randomly selected seed, only manage to infect one additional node

8.4 Halting Technique Analysis

The epidemiological model represents the spreading phase of multimalware outbreaks. The following approximate analysis of this phase establishes a lower bound on the diversity L needed by the halting technique summarized by Table 8.1.

We first clarify why hubs should be immunized in an inhomogeneous spreading network. When the infection probability is *small* and the malware spreading originates from a *single* randomly selected seed, a strategically placed node in the "core" of a monoculture contributes more to the spreading than a hub on the network's periphery does [81]. However, we study polycultures with a *maximum* infection probability equal to one and *multiple* widespread seeds per node type. Consider a hub with a large degree D on the periphery of a network. Since the S seeds with the same type as the hub are uniformly distributed over the network, one of the hub's neighbors could be a seed. When this seed infects the hub, the hub will again infect roughly D/L of its neighbors of the same type. We want to prevent this peripheral hub infection because D tends to be much larger than L and because any of the D/L infectious nodes can cause extensive malware spreading when the infection probability is one.

Since different malware spreading mechanisms result in distinct spreading patterns, it is essential to analyze malware outbreaks on spreading networks with arbitrary degree distributions. Let the nodes in a network be numbered from one to N and let node i have degree k_i, $i = 1, \ldots, N$. We consider the ensemble of random networks with an arbitrary but fixed degree sequence $\{k_i\}$ generated by the so-called configuration model (see [23, Sect. 13.2] for details). All networks have a mean degree $\langle k \rangle = 1/N \sum_i k_i$ and a mean-squared degree $\langle k^2 \rangle = 1/N \sum_i k_i^2$. Any network has L node types, with (approximately) N/L nodes of each type.

A *single-type component* is a subset of nodes of the same type such that there is a path between any pair of nodes in the set and such that it is not possible to add another node of the same type to the set while preserving this property. The two orange nodes in Fig. 8.1 constitute the largest single-type component. A single-type component is a *giant component* when its size is proportional to N/L. If a single-type component contains a seed, then all its nodes will be infected.

We study single-type components in a network to limit the overall fraction of infected nodes. Let this fraction be averaged over many model runs, where each run has $L \cdot S$ randomly selected seeds. The underlying network topology is the same for all malware types, since they are assumed to have the same spreading mechanism and the nodes of different types are uniformly distributed over the network. A particular malware type only infects a single type of nodes. Hence, malware of different types infects distinct subsets of nodes. Because each subset has N/L nodes, all subsets have the same fraction of infected nodes when averaged over many model runs. Consequently, the average fraction of infected nodes over all types can be analyzed by considering a *monoculture subgraph*, defined by all the nodes of an arbitrary but fixed type and the edges connecting these nodes. All other nodes and their adjacent edges can be ignored.

To limit the average fraction of infected nodes, we want to choose the diversity L such that the monoculture subgraph does not have a giant component. This subgraph has a mean degree $\langle k \rangle / L$ and a mean-squared degree $\langle k^2 \rangle / L^2$ for large N. Because the subgraph is contained in a random network generated by the configuration model, the subgraph has a giant component if and only if $\langle k^2 \rangle / L^2 > 2\langle k \rangle / L$ in the limit for large N [23, p. 456]. To prevent the formation of a giant component, we need $\langle k^2 \rangle / L \leq 2\langle k \rangle$ or, equivalently, we choose the diversity L such that

$$L \geq \left\lceil \frac{\langle k^2 \rangle}{2\langle k \rangle} \right\rceil. \tag{8.1}$$

The right-hand side of inequality (8.1) is large for inhomogeneous networks because k_i^2 is much larger than k_i for hubs. However, hub immunization reduces the lower bound. When the nodes with the largest degrees in the original network are immunized, we obtain a new network with $N' < N$ susceptible nodes and smaller node degrees d_j, $j = 1, \ldots, N'$. The new network is obtained by ignoring all immunized hubs and their adjacent edges because they no longer contribute to malware spreading. This network "pruning" affects the previously discussed monoculture subgraph. The new mean-squared degree $\langle d^2 \rangle = 1/N' \sum_j d_j^2$ and mean degree $\langle d \rangle = 1/N' \sum_j d_j$ should be substituted for $\langle k^2 \rangle$ and $\langle k \rangle$ in inequality (8.1) to determine the minimum needed diversity L.

Whether or not hubs in the original network are immunized to obtain a new network, the S seeds in a monoculture subgraph can spread over at most S components of this subgraph. These components are small in graphs without a giant component [23], leading to a small fraction of infected nodes. Inequality (8.1) shows a trade-off between the required number of node types L and the number of immunized hubs. If it is possible to generate many node types, then the fraction of immunized hubs can be reduced, making it possible to halt malware outbreaks on very large inhomogeneous networks.

8.5 Halting Technique Performance

We have seen that the hubs in a spreading network with inhomogeneous topology can be immunized to obtain a homogeneous network. If the hubs are not known, then acquaintance immunization can be used to protect most hubs [25]. Acquaintance immunization will be discussed in Chap. 9. Here, we apply the malware-halting technique to synthetic and empirical spreading networks with homogeneous topologies. Each network represents the worst-case spreading of S malware outbreaks per node type. While inequality (8.1) is only strictly valid for random networks in the limit of large N, the following NetLogo [46] simulations show that the lower bound determines the needed diversity.

8.5.1 Sparse and Homogeneous Networks

Wireless devices, particularly smartphones, can communicate via short-range wireless links such as Wi-Fi and Bluetooth links. In our first epidemiological simulations, different malware types copy themselves to new devices by opening wireless connections. Sparse and homogeneous proximity networks represent the spreading patterns. The NetLogo model generates a proximity network with an average node degree $\langle k \rangle$ by first placing N nodes uniformly at random on a square. An edge is then added between a randomly chosen node and its closest neighbor in Euclidean distance.

Table 8.2 The minimum number of node types needed to halt malware outbreaks on homogeneous proximity networks with 5,000 nodes and an increasing average node degree

Proximity networks				
Average node degree $\langle k \rangle$	5	6	7	8
Minimum needed node types L	3	4	4	5
Fraction of infected nodes	3.4 %	3.6 %	4.6 %	4.8 %

Each fraction of infected nodes is averaged over 500 networks with uniform random distribution of node types and seeds

More edges are similarly added until the network has the desired average degree $\langle k \rangle$. Self-loops and multiple edges between nodes are not allowed. Note that although handheld devices move over time, we only model short-term malware spreading assuming static networks. Wireless sensor networks stay fixed for long periods.

Table 8.2 lists the lower bounds on the needed diversity L, obtained from inequality (8.1), for proximity networks with 5,000 nodes and an increasing average degree $\langle k \rangle$. Each fraction of infected nodes is averaged over 500 networks with the same average degree and uniform distribution of node types, including $S = 10$ seeds per type. Only connected networks were evaluated, that is, networks with isolated subgraphs were ignored. The lower bound on the diversity L was the same for all evaluated networks with a given average degree.

While the deterministic epidemiological model causes all nodes to become infected in a monoculture ($L = 1$), less than 5 % of the nodes became infected in the diverse proximity networks, according to Table 8.2. Previously published simulation results and mathematical analyses of other network models confirm that small true or artificial diversity is sufficient to halt multimalware outbreaks on homogeneous and sparse networks [82].

We also analyze malware halting on a sparse network where the nodes represent email addresses and the links represent e-mail exchanges between the addresses. The network has 1,133 nodes and 5,451 edges. The largest node has degree 71 and the average degree is 9.62. While the network is slightly inhomogeneous, we forgo the immunization of large-degree nodes. The lower bound on the diversity is $L \geq 10$. Since the network is small, we assume only $S = 1$ seed per node type.

Ignoring the fact that email malware needs help from unknowing users to propagate, the simulations determined the fraction of infected nodes averaged over 5,000 random configurations of node types and seeds for increasing diversity $L = 10, 11, \ldots, 16$. The fraction of infected nodes decreases from 8 to 4 % when the diversity increases from 10 to 16. The additional decrease in the fraction of infected nodes is relatively small for diversity above the lower bound in inequality (8.1). Earlier reported simulation results for other networks [82] show similar modest reductions in the fraction of infected nodes for diversities beyond the minimum required value.

8.5.2 Dense and Homogeneous Networks

Consider the case where L types of random scanning malware spread over a complete network with N nodes of degree $k = N - 1$. There are L node types and N/L nodes per type. The types are uniformly distributed over the nodes of the network. Assume one seed per node type. Each seed has edges to the other $N/L - 1$ nodes of the same type. Together, the N/L single-type nodes form a star graph with the seed in the center. Since the seed will always infect all the peripheral nodes in the star graph, it does not help to increase the number of node types L as long as there is one seed per node type. All N nodes will still be infected. The only way to halt multimalware outbreaks is to use many more nodes types than there are malware types.

If there are M malware types, then $M \cdot N/L$ nodes will be infected. Hence, the diversity L needs to be proportional to N and the number of malware types M must be much smaller than N to prevent a large infection. This observation is in accordance with the diversity bound in inequality (8.1), which is equal to $L \geq (N - 1)/2$ for $k = N - 1$. More generally, consider an arbitrary path consisting of m edges in a dense network. The path's nodes are all of the same type with probability L^{-m} for $m \leq N/L$. We must have diversity $L \approx N$ to ensure that this probability is very small even for very short paths. As stated in Table 8.1, large artificial diversity is needed to halt malware spreading over homogeneous dense networks with many nodes.

Since it is not completely clear how much artificial diversity is obtainable with compilers utilizing diversification techniques [24], we cannot conclude that the halting technique is applicable to multimalware outbreaks with *dense* spreading patterns. However, Todd Jackson [83] and his colleagues convincingly argue that application stores can produce massive-scale software diversity. Furthermore, as we transition from Internet protocol version 4 (IPv4) to IPv6, topological scanning may become more popular than random scanning due to the huge number of unused IPv6 addresses.

8.6 Persistent Targeted Attacks

The term *advanced persistent threats* refers to attackers employing more or less advanced techniques to first learn about and then compromise selected computer systems without being detected, at least not for a long time [75, 76]. Examples of persistent threats are state-sponsored attacks on foreign commercial and governmental enterprises to steal industrial and military secrets. The attacks are often initiated by well-timed, socially engineered spear-phishing emails delivering trojans to individuals with access to sensitive information. Malicious email is leveraged because most enterprises allow email to enter their networks.

Persistent attackers frequently exploit OS or application vulnerabilities in the targeted systems. An attacker first develops a payload to exploit one or more vulnerabilities. Next, an automated tool such as a PDF or Microsoft document delivers the payload to a few users of a system. The payload installs a backdoor or provides remote system access, allowing the attacker to establish a presence inside the trusted system boundary. Finally, the attacker violates the confidentiality, integrity, or availability of the system to achieve his or her goals.

We shall see that large software diversity increases the time persistent attackers need to compromise systems, thus providing defenders with more time to detect the probing of their system defenses, collect information about the attackers, and deploy countermeasures to prevent major system breaches. As before, we divide the binary files implementing the functionality of a particular program into L roughly equally large classes such that all members of the same class share at least one exploitable vulnerability, while members of different classes have no common exploitable vulnerability. If a user and an attacker download the same program from an application store [24, 83], then the two downloaded files share an exploitable vulnerability with probability $1/L$. When the diversity L is large, the probability of a common vulnerability is small and attackers can no longer reliably analyze their own downloaded program files to exploit vulnerabilities in users' program files. (Note that the diversity L must be large even if the lower bound in inequality (8.1) is small.)

Directed attacks against specific computers running known programs become more difficult, as long as the attacker has no way of determining which specific binary is running on what computer. Since it is necessary to create security patches tailored to the different binary versions of the same program [24, 83], it becomes impossible for an attacker to reverse-engineer software patches by comparing a particular patch to the corresponding code on a user's computer because the patch and code are both unknown to the attacker.

8.7 Related Work

Miguel Garcia [78] and his colleagues have studied true diversity at the OS level by considering exploitable OS vulnerabilities published over a period of roughly 15 years. The authors carefully analyzed vulnerabilities in 11 different OSs to determine how many of these vulnerabilities occur in more than one OS. More than 50 % of the 55 studied OS pairs have at most one remotely exploitable common vulnerability. The low numbers of shared vulnerabilities for different OS combinations strongly indicate that true diversity is obtainable with off-the-shelf OSs. The authors also provide a good overview of related research on software diversity.

Jin Han [79] and his colleagues have shown that true diversity is available at the application level with off-the-shelf software. The authors analyzed over 6,000 application vulnerabilities published in 2007. About 98.6 % of the studied applications have substitutes, that is, applications that offer similar functionality, and the majority of the applications either do not have the same vulnerability or cannot be compromised with the same exploit code. Nearly half of the applications are officially supported to run on multiple OSs. Although the different OS distributions of the same application are likely to suffer from the same vulnerability, the attack code is different in most cases.

Work by Konstantinos Kravvaritis [84] and his colleagues supports the need for more software diversity in real networked systems. The authors reasonably assume that binary files with the same name are realizations of a single program; that is, the files may be different at the binary level but their functionality is identical. A client–server application collected executable program and library files from individuals who installed the client application on their computers. The client calculated the MD5 hash of each collected file and sent the hash to the server. Since the hash is unique for each different input file, the server could determine whether or not binary files with the same name were identical.

Kravvaritis [84] and his colleagues defined three metrics to measure the diversity of binary files with the same name. One metric, which estimates the probability of a successful targeted attack, is given by m/n, where m is the number of instances of the most frequent binary variant of a program and n is the total number of instances. The server collected 1,309,834 binary instances of 205,221 files with different names. For more than half of the files analyzed, the estimated chance of a successful attack is in excess of 50 %. The values of all three metrics indicate that the diversity of current software platforms is too low to significantly slow down targeted attacks. Hence, there is a real need for the large compiler-generated diversity discussed in this chapter.

Research by Pu Wang [85] and colleagues confirms that the number of giant components with nodes of the same type determines the extent to which malware of different types spread over diverse networks. The authors study the calling patterns of 6.2 million mobile phone subscribers to determine possible spreading patterns of malware attacking smartphone OSs. When a smartphone OS's market share is small, there is no giant component of the call network connecting most phones with this OS. Although the call network is connected, a subgraph of smartphones sharing the same OS is fragmented into many small and disjoint components [85]. The lack of large components on which different types of malware can spread explains the low observed saturation of malware in real mobile phone networks. Nevertheless, future malware epidemics are possible because two OS families currently dominate the smartphone market and more and more people buy smartphones.

Juan Caballero [86] and his colleagues have shown that the judicious use of true diversity improves the robustness of the Internet routing infrastructure against software vulnerabilities facilitating denial-of-service attacks, remote execution of system-level commands without authentication, and unauthorized privileged access. While the use of different software implementations from different code bases on different routers increases the network's overall robustness, it also increases the complexity and costs of network deployment and management. Artificial diversity, as suggested by Franz [24], is an interesting alternative to true diversity because the complexity and costs are much reduced.

Graph coloring is the assignment of colors to the nodes in a graph subject to a constraint [86]. Not surprisingly, a good coloring algorithm needs fewer colors to obtain adequate true diversity on a network compared to just distributing colors uniformly over the network devices, as done here. Because the best coloring algorithms necessitate central coordination to install the correct software on the different devices, these algorithms are best suited to slow-changing infrastructures managed by skilled personnel. Coloring algorithms are less useful when general users manage computing devices. The advantage of deploying application stores incorporating compilers with diversity engines is that adequate diversity is achieved with very little involvement from device owners.

8.8 Summary

While the Internet's numerous networks are diverse due to distinct configurations, firewall rules, anti-malware signature sets, intrusion detection, and router policies, many networks still have limited internal diversity, making them vulnerable to serious malware spreading. The multimalware-halting technique presented can halt outbreaks on these networks.

Advanced persistent threats represent a serious challenge to defenders of networked systems with very sensitive information. Our analysis shows that software diversity makes it harder to infect computing devices in these systems. Eventually, large-scale experiments will be needed to determine how to best deploy software diversity to make systems more robust to malware.