BELFER CENTER PAPER

Ending the Cybersecurity Arms Race

How Big Data and Network Monitoring Can Restore the Advantage to Defenders

Jim Waldo

Katherine Mansted

HARVARD Kennedy School
BELFER CENTER
for Science and International Affairs

PAPER
FEBRUARY 2019

Belfer Center for Science and International Affairs
Harvard Kennedy School
79 JFK Street
Cambridge, MA 02138

www.belfercenter.org

This work was made possible by a gift from Microsoft.

Layout and design by Andrew Facini

Cover image: Adobe Stock

Printed in the United States of America

BELFER CENTER PAPER

Ending the Cybersecurity Arms Race

How Big Data and Network Monitoring Can Restore the Advantage to Defenders

Jim Waldo

Katherine Mansted

HARVARD Kennedy School
BELFER CENTER
for Science and International Affairs

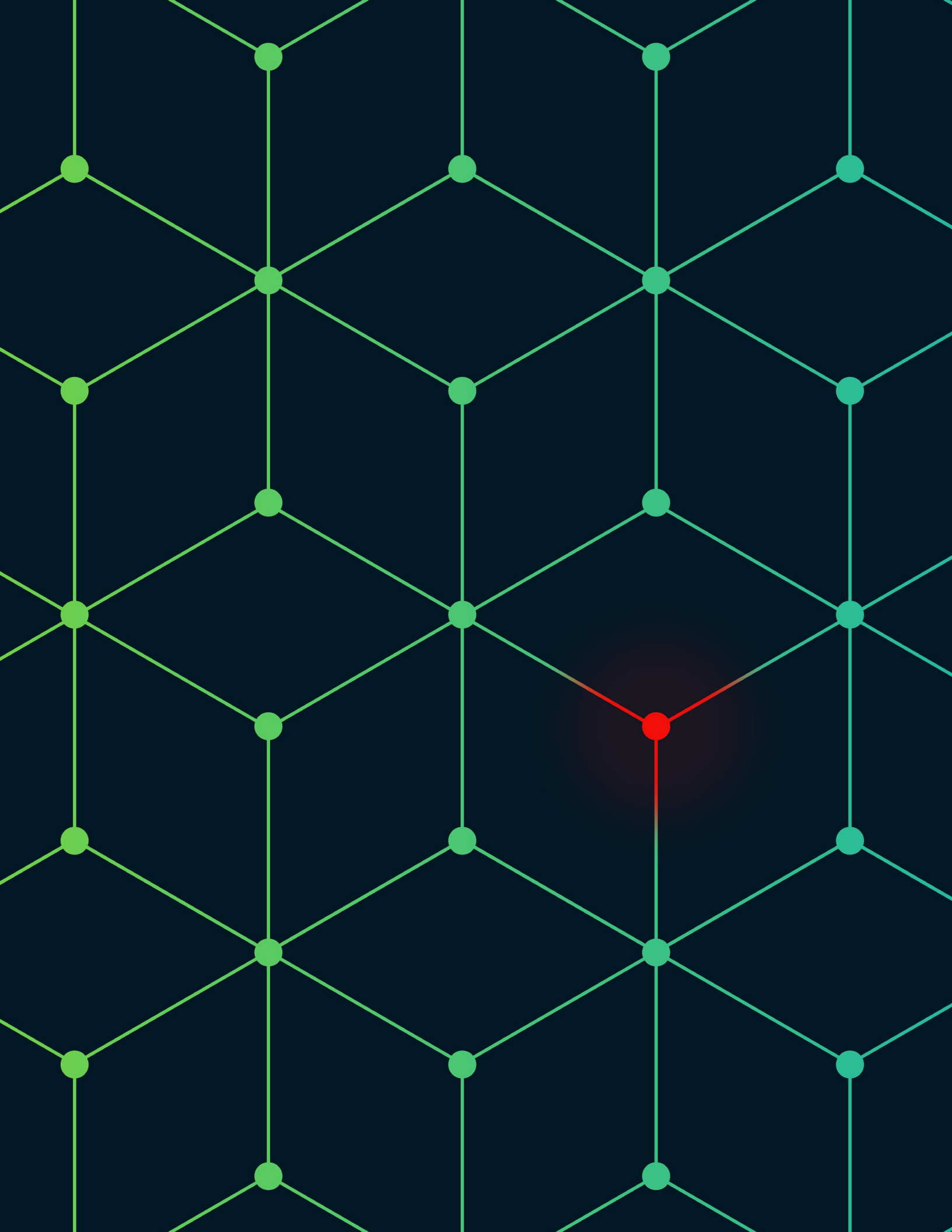
PAPER
FEBRUARY 2019

About the Authors

Jim Waldo is a Gordon McKay Professor of the Practice of Computer Science and the Chief Technology Officer of the John A. Paulson School of Engineering and Applied Sciences at Harvard University, and a Professor of Technology Policy at the Harvard Kennedy School. He teaches and does research in areas that are at the interface between technology, policy, the law, and ethics. He holds a Ph.D. in philosophy from the University of Massachusetts, Amherst.

Katherine Mansted is a Senior Research Officer at the Australian National University's National Security College, and a Nonresident Fellow at the Belfer Center. Her research focuses on emerging technologies and national security. Previously, Katherine practiced law as a commercial solicitor and served as a ministerial adviser in the Australian Government. She holds a Master of Public Policy from the Harvard Kennedy School.

Table of Contents



Executive Summary

Network security has always been something of balancing act between maximizing sharing and ease of use, and erecting barriers.

When computer networks first emerged, there were few limitations on what could be transmitted over them. However, after the world's first major network computer security incident—the Morris Worm of 1988—organizations began to retreat behind network-level firewalls and anti-virus software. Some defenders even tried to completely disconnect their networks from the outside world via "air gaps."

This paper argues that it is time to move beyond the security paradigm of separating networks, as epitomized by the air gap. Instead, network defenders should embrace an approach which allows sharing and connectedness, anticipates that adversaries will penetrate the network, and is able to detect, and ultimately eject those adversaries before they can do harm.

In Part 1, we trace the history of network security. While the Internet was designed for maximum flexibility, and security was initially managed at the level of each end-point in the system, this changed with the introduction of firewalls and anti-virus software.

In Part 2, we show how the use of these barriers then started an arms race between attackers and defenders. As defenders tried to filter out all malware using anti-virus software, attackers developed new ways of masking the malware they produced. As defenders built more complex and layered firewalls, attackers probed for new ways to penetrate those walls.

Part 3 then looks at the logical extreme of the barrier-based approach to network security, the air gap, in which a system is disconnected from the rest of the cyber world. However, air gaps do not solve the problems of network security. If using isolation to defend a network, defenders must be perfect—needing to anticipate and prevent all attacks—while the attacker need only find a single flaw in the defense.

The Stuxnet malware pointed up the limitations of an approach to network security that relies on isolating resources, while the Snowden leaks showed that even a perfectly separate network is vulnerable to insider threats.

Part 4 argues that an approach to security that emphasizes detection of an adversary and ejection once detected helps to stop the debilitating arms race in which, to this point, defenders have largely been on the losing side. With a "behavioral monitoring" approach to network security, the adversary must find a perfect strategy that blends into the normal use of the targeted system so well that their attack cannot be detected, and the defender need only find a weakness in that strategy. This approach therefore takes the advantage from adversaries and shifts it to defenders.

We conclude by explaining how, in practice, an organization might choose to implement a behavioral monitoring approach. Of course, this approach will still involve basic hygiene measures—such as installing updates and encrypting data at rest and on the wire. But this should be combined with monitoring and detection of anomalous behavior that may show that an adversary has gained access to the network. This monitoring may be conducted at the level of an individual network, but will be most successful when at least some infrastructure is outsourced to organizations that can invest more in both expertise and tools to do the needed anomaly detection. Additionally, these third-party organizations analyze much larger sets of data, have visibility into a wider variety of attacks, and can employ specialized experts in big data and machine learning, significantly increasing the effectiveness of behavioral monitoring in their environment. Thus, once again, network-connected systems can benefit from sharing and connectedness—rather than going it alone.

Introduction

On November 2, 1988, the idea of security on the Internet changed completely.

Prior to this day, security on the Internet (or any other network) was an extension of the security notions that had been designed for the shared mainframes and mini computers of the non-networked age. The network was seen as a way of transmitting packets of bits from one computer to another. Security was something that happened on the computers connected by the network, not in the network itself.

All this changed on November 2, 1988 when the first Internet worm infected an estimated 10% of the computers connected to the Internet. The Morris Worm, as it became known, spread by email and took control of infected systems by utilizing a buffer-overflow in the Sendmail program, widely used by Unix systems[1] for sending and receiving electronic mail. Spreading so quickly that it could not be stopped by the intervention of human system administrators, the Morris Worm replicated on an infected machine until it consumed all of the resources on that machine. It also sent itself to any other machine it could find.

The Morris Worm showed the underlying inadequacy of the computer-only approach to security in the emerging networked world. However, it was already too late to change the underlying communication protocols that were being used on the Internet. Instead, additional security was built on top of those protocols. The new strategy tried to detect packets in the network that contained dangerous content, and to block packets from being delivered in any but a very controlled way. The first of these approaches led to the development of anti-virus software, while the second led to the development of firewalls.

However, the use of anti-virus software and firewalls to protect computers connected to a network started an arms race between those trying to compromise networks and those trying to protect them. As the number of

1 Unix is an operating system now commonly used by servers. At the time of the Morris Worm, these systems were widely used in research, education, science and engineering.

computers connected to the network increased and the sophistication of the users of those computers decreased, attempts to isolate those computers from harm became more layered and more complex.

Some organizations attempted to "air gap" their networks, hoping that by isolating themselves from the outside world, they could deny attackers the opportunity to attack. An alternative approach, which we call "behavioral monitoring," sees monitoring the network for anomalous behavior as more important than isolating the network from any attack.

This approach understands that most attacks are actually the end result of long campaigns to penetrate, instrument, and then exploit a system, and that by detecting odd behavior in the network these campaigns can be stopped well before the adversary is able to capitalize on any discovered weakness. It is also enhanced by, and dovetails nicely with, the current trend for companies and governments to move computing resources to "the cloud"—since behavioral monitoring operates best at scale.

Part 1. A short history of network security

Originally, the security of networked computing systems was an extension of the mechanisms used for securing single computing systems that were used by multiple users. This model had significant impact on the design of the network itself, allowing the network to be highly flexible but leaving security to the end points of the network. By the time this approach was shown to be problematic, changing the underlying network was practically impossible, leading to attempts to layer security on top of the underlying protocols. Understanding this history is important to help us understand the security situation we find ourselves in today.[2]

2 We use "computer" to refer to a single machine, and "network" to refer to what connects computers. We use "system" to refer to both the network, and the computers that network connects, and which are under a single administrative entity.

Security basics

Security on an individual machine

Computer security on a single machine has always been based on the notions of authentication and authorization.

- **Authentication** answers the question "who wants access?". The identity may be that of a person but might also be of a program or some combination of a person, a program, or even a particular role. Mechanisms used for authentication were first used when multiple people shared a single computer. Such mechanisms include passwords (something you know), devices (something you have), and biometrics (something you are).
- **Authorization** answers the question "does the requestor have the privileges to access the resources requested?" Authorization is often simply a list of those (authenticable) entities that are allowed access to resources, although there can also be authentication based on group membership or role.

This sort of security is needed even if a computer is not connected to a network. Establishing your identity by logging in (authentication) allows you to access some set of resources but not others (authorization). We see this today with separate logins for each family member on a shared home personal computer, or when we log in to a shared server that allows us to access files owned by many different people. When the question of security arose in the early days of networking, this basic model, based on authentication and authorization, was extended to the new environment.

Early network security

There is a persistent belief that the Internet[3] was designed without thinking of the security implications of the network.[4] On this view, the engineers who built the original Internet all knew each other, never understood how large the network was going to get, and were busy getting the basic functionality to work. They simply shipped the technology before thinking about the security of the system, and we have been paying for their naivety or haste ever since.

While it may be that the original designers of the Internet all knew each other, and never guessed how large or important the network would become, the notion that they hadn't thought about the security of the network is demonstrably false. The fact is that in *End to end arguments in system design*,[5] as close to the founding design document of the Internet as one can get, adding security to the network is explicitly discussed *and firmly rejected*. Security was not added to the Internet not because the original designers didn't think of the issues around security; security was not added to the original design of the Internet because the engineers determined that it would be a mistake to do so. Examining their reasoning is instructive, both in clarifying the historical record and in understanding the overall problem of network security.

End-to-end arguments

The basic argument against adding security to the underlying network is fairly simple. It begins by taking the model of security that had been used on a single (often shared) machine and assuming that an extension of that model will be used on the network. The individual computers joined by the network will each need to authenticate the entity requesting access and then see if that user has the authorization needed for the access.

3 When we talk about the Internet, we will mean a network based on the TCP/IP set of protocols.

4 For a recent example, see The Economist. April 18, 2017. The Myth of Cyber-Security: "[Software vulnerabilities] are compounded by the history of the internet, in which security was an afterthought."

5 J.H. Saltzer, D.P. Reed, and D.D. Clark. 1984. "End-to-End Arguments in System Design," ACM Transactions on Computer Systems, 2(4).

Authorization decisions are local, in that only the computer on which a resource resides can know the restrictions on who or what can use that resource. This in turn means that authorization needs to be determined at the endpoints of the network, no matter what security is built into the network itself. Further, even if authentication occurs at the level of the network, it needs to be made again at the endpoint, just to ensure that the authentication was made correctly. Since these decisions have to be made at the endpoint, also making them at the level of the network means that the same work is being done twice. This puts extra work and complexity in the network, and doesn't save the endpoints from having to do the same work. But if we do the work at the endpoint and not at the network level, the security is not compromised.

The core of the end-to-end principle is that any work that will need to be done at the endpoints of the network should not also be done at the level of the network. While security was one example of this kind of work, other examples include error correction and packet re-transmission. The end-to-end principle pushes the network to be as simple as possible—all the network does is deliver packets. The power of this simple model should not be underestimated. By keeping the network simple and only delivering un-interpreted bits, the Internet has come to be used in ways that were never imagined by the original designers. But it does mean that much of the functionality we want—including security—was pushed to the endpoints.

Problems with endpoint security

While moving all security out of the network and into the computers connected by the network kept the design of the Internet simple and enabled that design to be used in new and innovative ways, it also is the root of a number of problems. These are the problems that have brought us to the security worries that we have today.

One of the first of these problems has to do with authentication, one of the twin pillars of the security model. Unlike the case of a single machine, network-wide authentication requires that there be some network-wide

notion of identity, or that the machines using the network have identities at all endpoints. When the Internet was reasonably small, it made sense to have login identities on all of the connected machines. Within a single corporate or educational organization, there is often a single authentication service for all those who are part of the organization. But as the Internet became what it is today, there was no single source of authentication for all users. (Hence the famous *New Yorker* cartoon: "On the Internet, nobody knows you're a dog."[6])

Authentication can be thought of as the dual of the notion of attribution—a term popularly used in current cybersecurity debates to refer to identifying which state, or other entity, is responsible for a particular cyberattack. If we required authentication for all of the messages sent over the network, we would have a form of attribution as well. Authentication is still done locally (for example, you log into your PC using a password, or your smartphone using your thumbprint to establish your identity). It is also done on many remote sites, for example when you log in to an on-line retailer or streaming service. While this could be extended to all computers on a network, but it would be complex and error prone, as users could, if following best practices, have a different identity on every machine on the network. It would be possible to build a system where users are given a single identity over the network in some distributed (often federated) fashion. The basics of the theory behind such an authentication system has been well understood for some time[7], and systems based on these principles have been implemented in a large-scale system.[8] But doing something like this at the scale of the current Internet is not something that anyone has attempted.

Another problem is the assumption that the security of each endpoint is independent of the security of other endpoints. This is generally not the case; information stored on one endpoint can help in the compromise of another endpoint, leading to contagion. If the security of an endpoint can compromise the security of another, then the system is only as secure as

6 https://www.art.com/products/p15063499994-sa-i6847752/peter-steiner-on-the-internet-nobody-knows-you-re-a-dog-new-yorker-cartoon.htm

7 See, for example, Butler Lampson, Martin Abadi, Michael Burrows, and Edward Wobber. 1992. Authentication in Distributed Systems: Theory and Practice, Proceedings of the Thirteenth ACM Symposium on Operating Systems Principles.

8 See https://web.mit.edu/kerberos/ for the most commonly used such implementation.

the weakest endpoint. If it held, the independence assumption would be a strong reason to adopt endpoint security; it means that the compromise of a particular machine or region of a network only impacts that machine or region. However, the interdependence of most networked machines means that the compromise of any machine will lead to the compromise of many other machines on the network. The independence assumption does not hold. In any large network, there will be machines that are relatively easy to compromise, and if those machines can be used to spread the contagion to other machines on the network, the whole network can become infected. Further, as machines become more and more likely to be administered by their users (rather than system administrators trained in security) the level of security on the average device attached to the network has gone down.

Finally, endpoint security works well when the community connected by the network is small and there are social mechanisms that can be used to reinforce the security. This was certainly the case in the early days of the Internet, when that network connected a group of academic and industrial researchers who all shared a value system, were known to each other, and who gathered together regularly for meetings or conferences. Intentionally violating the security of the network was counter to the community norms, and the users of the system were technically sophisticated enough to know how to secure their individual machines.

Despite its limitations, endpoint security was kept long after the Internet grew beyond this small community. Networks were used to connect the employees of large corporations, and the Internet was used to connect those networks. While still insignificant in terms of the number of users when compared to the Internet today, the Internet in 1988 still seemed like a safe place for researchers and technologists. All this changed with the advent of the Morris Worm.

The end of endpoint security

The Morris Worm

On November 2, 1988 the Morris Worm infected, in a remarkably short period of time, a significant number of the computers that were connected to the Internet.

Written by Robert T. Morris, a graduate student at Cornell University, the Morris Worm was released from the Massachusetts Institute of Technology. It is widely accepted that Morris did not have any malicious intent, but released the worm as an academic experiment. However, the worm spread significantly faster than he had anticipated.

The Worm used several attack methods. In addition to exploiting known vulnerabilities in software based on the Unix operating system, it also used 'brute force' to guess user passwords since, in 1988, many users used easy-to-guess passwords, such as common dictionary words.[9] In this way, it demonstrated that networks face security threats not just from unauthorized people, but from unauthorized programs. The Morris Worm was self-replicating, and would scan for new machines to infect automatically—without human assistance. Since it could infect single machines multiple times, the worm also acted as the Internet's first significant denial of service attack.

While it did not modify or destroy systems or data, the Morris Worm impaired, and ultimately halted, infected computer's performance, as their systems became overloaded by trying to run many copies of the worm program.[10] Within hours of its release, the Morris Worm is estimated to have infected about 10% of all computers connected to the Internet—including computers at most of the US government and private research centers.[11]

9 J. Reynolds. 1989. "The Helminthiasis of the Internet" IETF Network Working Group. Paper 1135. Available: https://tools.ietf.org/pdf/rfc1135.pdf

10 Eisenberg et al. 1989. "The Cornell Commission: On Morris and the Worm." Communications of the ACM, 32(6).

11 United States General Accounting Office (GAO). June 12, 1989. "Computer Security: Virus Highlights Need for Improved Internet Management". Report to the Chairman, Subcommittee on Telecommunications and Finance, Committee on Energy and Commerce, House of Representatives. Doc ID: GAO/IMTEC-89-57. Available online.

At this time, the total number of computers connected to the Internet was fewer than 100,000.[12] Although it was mostly eliminated within two days, the Morris Worm may have cost the US economy up to $10m.[13]

To defend against the Morris Worm, or to eliminate it from machines on their network that had been infected, companies and regional networks detached themselves from each other, and from the NSFNet backbone (then the main Internet connection). This form of network isolation allowed the damage that was done by the Morris worm to be localized, and ensured that if the worm had been removed from a network it would not be re-introduced by being sent from someplace else.

Firewalls and anti-virus software

The Morris Worm made it clear that relying on the end-to-end principle, at least with respect to security, posed a significant risk to organizations that depended on computing infrastructure that is connected to the Internet. Since the Morris Worm exploited a "zero-day" (a previously unknown vulnerability) in the email system, the standard endpoint security mechanisms of authentication and authorization were not sufficient, since the Worm had exploited a vulnerability in the underlying software that bypassed any mechanisms for authentication or authorization. What was needed was some defense that stopped such attacks before they even reached the endpoint machine.

Trusting the Internet was no longer an option. But the number of machines that were connected to the Internet, using the TCP/IP protocol, also meant that it was infeasible to change the underlying network architecture. To secure networks, something would have to be layered on top of the network rather than being added to it.

The reaction was a change in the way that networks were defended. Rather than letting network packets with unknown content reach the endpoint

12 Tim Berners Lee. November 1, 2013. "How a grad student trying to build the first botnet brought the internet to its knees." The Washington Post. Available online.

13 GAO report, above.

machines, as is required by endpoint protection, the dominant mode of defense became a combination of filtering out suspect content and isolating the network inside some boundary that could keep malicious actors (and code) out. As Tim Berners Lee has explained:[14]

> *Before Morris unleashed his worm, the Internet was like a small town where people thought little of leaving their doors unlocked… The Morris Worm destroyed that complacency.*

The problem then became deciding what content was malicious, and how to establish the boundary between the inside of the network and the outside. Specialized software was developed for each of these tasks, which are now known as firewalls and anti-virus protections.

- Firewalls are components of a network security system designed to limit the traffic into and out of a part of the overall network. Generally implemented in the network switches themselves, firewalls are configured with a set of rules that determine what traffic can flow into or out of that switch.[15] While the technology used in firewalls has evolved (allowing, for example, firewalls to be configured based on the content of the packets being allowed through), the security theory behind firewalls has remained the same. The whole notion of a firewall is to insulate a part of the network that is somehow trusted from the rest of the network, which is not. Like a walled town during the middle ages, those computing devices inside the firewall are free to interact with each other (subject to the existing endpoint security mechanisms of authentication and authorization). Those outside the wall are allowed to enter into the trusted area only if they can pass certain rules, showing that they have no bad content or intent. Like walled towns, firewalls have as their weakest link the "gates" they provide, that is, the network traffic that is allowed through by the firewall. Hence firewalls are often paired with anti-virus software.

14 Tim Berners Lee. November 1, 2013. "How a grad student trying to build the first botnet brought the internet to its knees." The Washington Post. Available online.

15 The configuration takes the form of a set of security rules; these rules can limit the sources of data (for example, only network traffic from particular machines will be allowed through the firewall) or only to a particular destination (for example, only traffic going to a certain set of ports, like those used for http traffic).

- **Anti-virus software** are programs that scan the network traffic that is let through the firewall looking for patterns that have been identified with malware (a portmanteau of "malicious software"). Initially, anti-virus software focused on identifying known viruses, but has evolved to also detect a variety of malware. If anti-virus software flags content as exhibiting content or characteristics consistent with malware, it will perform actions ranging from informing the user, to quarantining or simply deleting the suspicious content. Initially, anti-virus software primarily relied on digital 'signatures' to match the content it scanned with known malware. As attackers have become more sophisticated (for example, by modifying or concealing code to prevent malware from being recognized) detection of previously unseen malware is increasingly also facilitated by use of behavioral analysis.

Part 2. An arms race develops

Following the widespread adoption of firewalls and anti-virus software, much of the history of network security can be seen as an effort to control what can move in and out of a network. This history is also a tale of how attempts to fully control who has access to resources in a network are defeated by adversaries who, when blocked in one way, find another entry point to exploit, creating both an industry for cybersecurity experts, and something of an arms race between attackers and defenders.

The limitations of firewalls and anti-virus software

A few examples can show the debilitating arms race that has developed between offense and defense:

- Some types of firewalls simply block certain types of traffic from entering the network, such as request for anything other than content from a web site. Such blockages rely on the wanted traffic arriving on a predictable network port (such as port 80 for web traffic). The assumption was that requests to a web server would be harmless. In response, attackers started sending content that could exploit weaknesses in those web servers to begin their attacks.
- Anti-virus software began by assembling known "signatures"[16] of malware, and comparing incoming network communications to those signatures. In response, attackers began making minor changes to the malware that would change the signature, but not the functionality of the malware. Since the signature was changed, the anti-virus software would let the malicious content through.
- Rather than trying to identify malware by its signature, some anti-virus programs evolved to running incoming content in a protected 'sandbox,' which isolates that content from the receiving

16 The signature of the content is a cryptographic hash of the content, which reduces the content to a probabilistically unique, fixed length bit-stream.

network, and then watching to see the result of running the content. Only when running the content in the sandbox is determined to be harmless is the content delivered to the network. In response, attackers have begun? began writing malware that either: (i) delays its malicious actions, or (ii) checks to see if it is being run in a sandbox and, if so, does nothing until it is delivered.

The problems of virus detection

In addition, there are some inherent flaws in anti-virus software. With any anti-virus software, there is a risk of "false positives," which can have serious consequences for the performance of the network.[17] For example, if an anti-virus program automatically deletes or quarantines suspect files, a false positive can render an operating system or certain applications unusable. Moreover, anti-virus software is ineffective against new, or previously unseen viruses–most notably, those which exploit zero-day vulnerabilities.

Additionally, the types of bad actors and their purposes for distributing malware has evolved over time. Earlier malware was more susceptible of detection, because its purpose was often to cause immediate mischief or inconvenience. Increasingly, malware is difficult to detect by design, since it might be used to silently recruit a machine to a 'botnet', or to secretly exfiltrate data from a network.

Hard shell, soft core

The approach of keeping bad actors or bad content out of the network, but allowing free access within the network, is sometimes referred to as the "hard shell, soft core" approach to security. It allows high degrees of sharing (within an organization's network), but also means that a penetration of the network by an adversary opens the entire network to that adversary. Adversaries have become adept at compromising the "weakest link" in a system, and then moving laterally to their target. Additionally, the hard

17 See for example: https://www.engadget.com/2010/04/21/mcafee-update—shutting-down-xp-machines/; https://www.cnet.com/news/flawed-symantec-update-cripples-chinese-pcs/.

shell, soft core approach—even if completely successful—does not protect against insider threats.

Consequently, defenders have modified the hard shell, soft core approach over time: they have applied the approach of building network walls to smaller and smaller sets of machines within what had previously been a trust zone. The idea is to ensure that a compromise of one walled set, either by an outside adversary or some insider, will not cause a compromise of the whole. However, this approach complicates and restricts the kinds of sharing available to users of the system, complicating both the system and the ability of the users to get their work done.

This approach also leads to much greater complexity in the overall system, as there is no longer a strict distinction between "inside" the network and "outside." Instead, there are layers of defense, and being inside one part of the defensive walls did not allow access to other, more protected parts. Each of the sections of the network now need their own rules defining who, or what, can access this part of the system. Like any sort of software, the additional complexity of these rules means that there is additional likelihood of there being a bug in the rules, allowing attackers access.

Part 3. Air gaps

One approach to the cybersecurity arms race is to attempt to completely separate the network to be protected from the outside via an "air gap." An air gap is the most radical form of a firewall possible. Such an approach simply disconnects the network to be protected from any other network; it is a firewall with no network gates through the wall. The theory is that, since there is no connection between the protected network and the outside world, no adversary originating from the outside world can penetrate this set of resources, as there is no path for the attackers to take. Certainly, the inability to move data or programs in or out of a system through a network makes many forms of attack more difficult. However, air gaps are not a panacea, and indeed can be counterproductive.

The problems with air gaps

An air gap strategy assumes that the set of resources being protected is completely self-contained. But no system, no matter what measures have been taken to isolate it from the network, is completely safe. There is always a need for some data to be passed into and sent out of a system if that system is to do anything that is useful.

If data is entered into the system, then the data entry points are a connection between the air-gapped network and the outside world. In an industrial plant, the compromise of a sensor used to input readings into a closed network provides a path for attackers to begin a compromise. If software is updated, the updates are a connection between the air-gapped network and the outside world.

When an air-gap is thought of as forming an impenetrable barrier, it often creates a false sense of invulnerability. The belief that the network being protected is completely disconnected from the outside world actually obscures awareness of the connection points that inevitably remain. Any of those points at which information, either as data or as software, passes in and out of the network is a potential vector for attack. By relying on the separation of the network from the rest of the world, all that is accomplished is to obscure where information passes into and out of the system and, critically, where and when the separation is violated. Air-gapping a network also makes information sharing more difficult, which often leads users of the network to find new and unexpected ways to circumvent the network protection, introducing new vulnerabilities.

Two of the most damaging system breaches of the past decade show the limitations of an approach like air-gapping. Both the Stuxnet cyberattack on the Iranian nuclear plant at Natanz and the leak of classified National Security Agency documents by Edward Snowden would have occurred if the networks in question were fully air-gapped (indeed, the Iranian facility was protected in that way). How these attacks happened are instructive of the security limits of air-gapped networks.

Stuxnet and the complete separation fallacy

One of the most famous cyberattacks to date, and the most prominent example of an adversary penetrating an air-gapped system, was the use of the Stuxnet malware to disrupt an Iranian uranium enrichment plant at Natanz. The systems used in the plant were isolated from the external world, and the plant itself was heavily guarded. Despite these precautions, the Stuxnet malware was introduced. The vector of introduction was the computers used by third-party technicians involved in programming and updating the industrial controllers used in the facility.[18] These computers themselves were infected by a wide-ranging attack, including dissemination of infected USB memory sticks, designed to spread the basic Stuxnet code to as many computers that would be used at the air-gapped plant as possible. The Stuxnet code looked for the particular software that was used to program the industrial controllers used for the centrifuges at Natanz, and if that software was not found, it did little but look for other computers to infect.

Once the Stuxnet malware determined it was inside the Natanz plant, it propagated until it infected the centrifuge controllers: and then delivered its payload. In particular, the malware, both caused the centrifuges to malfunction, and told the monitoring software that all was well—so that the attack could progress unseen. The impact that the attack had on the plant, and Iran's nuclear program more broadly, is difficult to determine, but the fact that Stuxnet was able to find its way into a protected network showed that even the best protected networks can be breached.

Snowden and the insider threat

The Natanz breach shows just how difficult it is to fully separate a network from the outside world. Even in cases where the network separation is technically complete, users of the network who are inside the perimeter pose a risk. The Edward Snowden leaks are perhaps the most famous example of the threat that "insiders" can pose to even the most secure

18 Symantec Security Response: W32.Stuxnet Dossier Version 1.4 (February 2011) https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf

networks. While there are few networks as protected as those of the US intelligence agencies, no technical protection can fully protect against the insider threat. Insiders like Snowden—then a National Security Agency contractor— are "authorized users," in the sense that they have authority to access some part of an organization's system. Snowden was able to collect, exfiltrate, and leak a huge amount of information using the access he was granted, in what a former Director of National Intelligence called the "most massive and damaging theft of intelligence information" in US history.[19] Even if the periphery of the National Security Agency's network was impenetrable, the network lacked the instrumentation to detect that an insider had exceeded his level of authorization to amass this scale of information.

Snowden was an example of a malicious insider—in the sense that he intended to do what he did. Non-malicious insiders can also subvert the protections afforded by air-gapping a network. Authorized users can compromise network security by negligence, accident, or because they have been socially engineered by a third party. Indeed, one of the features of modern computer networks (unlike the early networks discussed in Part 1) is that most users are *not* technical experts. Technical solutions, like air-gapping, work best when users understand the security model being used. Today's non-technically sophisticated users are more likely to act in ways that inadvertently subvert protection. For example, research from behavioral science suggests that if security features are too difficult to use, or obstruct productivity in the workplace, they are often bypassed by users inside the network. If connections between different networks within and outside of the organization are bottlenecked or made non-existent by an air-gapping approach, or the broader Internet is walled off, users are likely to find novel (and unmonitored) ways of making these connections. This will make their organization even more vulnerable.

19 Testimony of Director of National Intelligence James R. Clapper, US House of Representatives Permanent Select Committee on Intelligence. Worldwide Threats Hearing (February 4, 2014).

Part 4. Behavioral Monitoring

A newly emerging trend in network security begins by rejecting the possibility of keeping an adversary out of a network, and placing the emphasis on instead monitoring inside the network. This new behavior-based approach assumes the use of some techniques to keep adversaries out, but then adds behavioral modeling to the traffic in the network. What is most significant about this new approach is the change in the goal of the endeavor. Rather than trying to separate the networked systems from intruders, the new approach assumes that the adversaries will, at least in some cases, breach the defense and lodge themselves in the network. The goal of the network barriers (and all systems require there to be some barriers) is to block some of the adversaries' advances. But the critical assumption is that these barriers will not be completely impenetrable, and will, at some point, fail. This approach to security concentrates on detecting when these failures occur, so that the adversaries can be removed from the system prior to doing harm. The physical analogy is that the walls around a city are not enough to thwart an attack. The walls are crucial part of the defense, but the city also needs a police force within the walls to catch those adversaries who penetrate the walls.

Behavioral monitoring basics

Pattern recognition

The core idea behind behavioral monitoring tools is to identify characteristic patterns exhibited by the traffic on any network. These patterns can be learned by the system using the techniques developed by those engaged in big data analysis and machine learning, and then the network can be monitored to watch for unusual behavioral patterns. Such behaviors can range from software installations to large data transfers to users logging in to systems that the users do not often use. Some anomalous behaviors will simply be flagged, allowing human users to judge if further action is required. Other anomalies will be automatically blocked or stopped, as

they are characteristic of actions that needs immediate attention. A user logging in to a system that the user does not generally use might be an example of the first sort of anomaly, whereas the transfer of a large amount of data to a single server within the network (often a prelude to exfiltrating that data) or connecting to an unknown server outside of the network (often the first step of exfiltration) is an example of the latter.

One of the insights that allows this approach is that few cyberattacks have the immediacy of the Morris Worm. Instead, most of what are reported as cyberattacks against organizations are in fact the end result of long campaigns in which the attacker gains access to the organization's network, scouts for interesting content or resources to attack, and establishes infrastructure inside the network, before they execute their malicious goals.[20] If the campaign can be detected and interrupted before the final attack, little or no harm will be done by the attacker.

Side effects of a statistical approach

Anomaly detection is inherently probabilistic, which means that systems are subject to both false positives and false negatives.

- A false positive is when the system flags (or stops) a behavior because it is seen as anomalous when in fact the behavior is an intended use of the system by an authorized user. Such false positives can be irritating, since it blocks the user from accomplishing an action. The system needs to be tuned to understand such uses, and there needs to be a way for the blocking of such actions to be manually overridden.
- False negatives are, in many ways, more detrimental. False negatives are situations that the behavioral tracking system fails to flag as anomalous activities but that are, in fact, the actions of an adversary who has infiltrated the network. Given the probabilistic nature of behavioral anomaly detection, the possibility of false negatives is always present; the goal is to tune the system to minimize the risk

20 For one take on this staged approach, see https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html.

that those behaviors that are not flagged are also those that are least likely to cause real harm.

Generally, these systems can be tuned over time. The more data the detection system has seen, the more sophisticated its ability to distinguish expected and anomalous activities. The number of false positives can be decreased by allowing a wider range of variation in the expected behaviors, but this may increase the number of false negatives. Additionally, the range of expected variation can be decreased, lowering the number of false negatives, but the number of false positives might be increased. Changes in an organization's procedures can cause the behavior on the network to change, and it will require a period of retraining and re-tuning the system to minimize false positive and false negative inaccuracies.

This is similar to efforts to block spam email. Most spam blockers are also based on machine learning algorithms that are trained to detect spam. Some spam gets through even the best of these systems (false negatives) and some legitimate email is blocked (false positives). But over time the systems become more precise, even when those who send spam work to generate email to thwart the detectors. Much of the same process is applied with various forms of malware and network intrusion detection.

Data superiority

Ultimately, the behavior-based approach favors those who have the most data, both on what the normal operation of their system is and on past and current attacks. Attacks are, fortunately, rare events. Establishing a pattern that characterizes an attack can be done more precisely if there are multiple instances of the attack in the data used to train the algorithms protecting the network. The more information that can be pooled about these threats, the better the training set that can be used to hone the algorithms for detecting the threats, and therefore the better those algorithms will be.

Implementing behavioral monitoring

First, it is important to emphasize that the introduction of behavioral monitoring should be seen as an addition to, rather than a replacement for, basic security measures that an organization is already taking. The introduction of behavioral modeling for network traffic does not completely replace the need for firewalls and anti-virus software. What it does do is eliminate the need for those measures to be perfect. Watching the network for signs of a breach means that we can balance the effort to keep the adversary out with the need to allow the users to be productive in their work.

Baselines

Products that enable behavioral monitoring exist, and new ones are being introduced at a rapid pace. Introducing these to the security regime protecting a system must be done early, so that the determination of a normal baseline can be established. While some anomalous behaviors can be detected using the history of attacks across all of a product's customer base, the important behavioral baseline is the one that is normal for the network being protected.

In-house and outsourced approaches

When adding behavioral monitoring to a system, and organization will also need to determine if it is best to do so in-house or outsource the work to a third party. The expertise needed to distinguish between suspicious anomalies and false positives requires a different set of skills than those needed to install anti-virus software or configure firewalls. Finding a trusted third-party provider can speed the transition to this enhanced defense.

An alternate form of outsourcing is to move part or all of an organization's computing and network infrastructure into a cloud provider that has developed expertise in this form of defense. Cloud providers already enhance the security of their tenants by ensuring that updates and

patches are installed in a timely fashion. The security of the cloud is also entrusted to teams that generally have greater experience and expertise in security than those that are available to cyber security providers that run security for organizations at a smaller scale. But the move to behavioral monitoring introduces new advantages to those who can move their infrastructure to the cloud. The main benefit is that cloud servers provide services to many clients. As mentioned, behavioral monitoring is an inherently probabilistic endeavor. Cloud providers have a very large set of machines, applications, and data which they can monitor to learn to identify anomalies. This scale makes even rare security breaches more common and more likely to be detected.

Conclusion

Since the Morris Worm of 1988, computing systems that have been connected to the Internet have attempted to secure themselves by detecting malware using anti-virus software, and isolating themselves from attacks using network firewalls. This has led to a thirty-year arms race, in which defenders have built ever more complex ways of keeping attackers out of their systems, and attackers have found new ways of avoiding these defenses. The extreme version of the firewall is the air-gapped network, which is completely disconnected from the outside world.

But even this has not proved sufficient to keep determined adversaries from compromising these systems. No system is ever completely static, either in the programs it runs, the data it contains, or the users that it serves. A system that was static in all of these ways would be useless. But all of these aspects of a system provide entry points for an adversary.

What is emerging is a new paradigm for cyber security, along with a new model for how to defend. Rather than trying to ensure that an adversary cannot enter a network-connected system, the new paradigm watches the behavior of the system. Using new techniques pioneered in big data and machine learning contexts, the new model looks for anomalous behavior inside of the network, and flags such behavior for investigation. The center of the idea is that a dedicated adversary will be able to penetrate any defense, so security requires watching for such a penetration and responding quickly when it occurs.

This paradigm changes cyber security to something that an organization does rather than something that the organization has. It is an activity that is constant. On this model, cybersecurity is more like treating a condition than it is like curing a disease. This reflects the fact that cybersecurity differs from other forms of risk management in that it is played against a sentient opponent. As the opponent changes approach, the defense must react. What is needed are the tools to inform defenders of a breach so that a proper response can take place.

Detection of anomalous behavior is never guaranteed, but can become more likely if the defenders maximize the data set used to train their detection algorithms. Like spam detectors that use a similar technology, the more data an organization has, the more likely it is that the organization can detect unusual activity that may signal a breach. This need for data, along with the specialized expertise needed to develop these detection systems, may require the outsourcing of the behavioral monitoring. In some cases, this will also argue for moving as much of a system to a shared environment such as a cloud environment as possible., since the cloud provider may have staff with the appropriate expertise and certainly has a larger data set from which to learn.

Belfer Center for Science and International Affairs
Harvard Kennedy School
79 John F. Kennedy Street
Cambridge, MA 02138

www.belfercenter.org

Printed in the United States of America