Reducing Attack Surface with Application Control

Version 1.4
Released: February 5, 2014

The Double-Edged Sword

The problems of protecting endpoints are pretty well understood. As we described in The 2014 Guide to Endpoint Security[1], you have stuff (private data and/or intellectual property) that others want. On the other hand you have employees who need to do their jobs and require access to that private data and/or intellectual property. Those employees have sensitive data on their devices, so you need to protect their endpoints to protect that data. It's not like this is anything new. Protecting endpoints has been a focus of security professionals since forever — with decidedly unimpressive results.

Organizations have spent tens of billions on endpoint security products and services. Yet every minute more devices are compromised, more data is stolen, and security folks keep having to answer senior management, regulators, and ultimately customers on why it keeps happening.

Why is protecting endpoints so hard? It can't be a matter of effort, right? Billions have been spent on research to identify better ways to protect devices. Organizations have spent tens of billions on endpoint security products and services. Yet every minute more devices are compromised, more data is stolen, and security folks keep having to answer senior management, regulators, and ultimately customers on why it keeps happening.

The lack of demonstrable progress comes down to two intertwined causes. First, devices are built using software that has defects attackers can exploit. Nothing is perfect, especially not software, so every line of code presents an attack surface. Second, employees can be fooled into taking action (such as installing software or clicking a link) that enables attacks to succeed.

These two causes cannot really be separated. If the device isn't vulnerable, nothing an employee does should result in a successful attack. And likewise, if no employee allows delivery of the attack/exploit code by clicking things (or navigating to the wrong pages), vulnerable software is less of an issue. So if you can disrupt either aspect your endpoints will be far better protected. Of course this is much easier said than done.

This digs into the good and bad of application control (also known as application white listing) technology, talking about how application control can stop malware in its tracks and mitigate the risks of both vulnerable software and gullible users. We address head-on the perception issues facing endpoint lockdown, which cause many organizations to disregard the

[1] https://securosis.com/research/publication/the-2014-endpoint-security-buyers-guide

technology as infeasible in their environments. Finally, we discuss use cases where application control makes a lot of sense and how it can favorably impact security posture, both reducing the attack surface of vulnerable devices and protecting users from themselves.

Accelerating Attacker Innovation

We mentioned the billions of dollars spent on research to protect endpoint devices. It is legitimate to ask why these efforts haven't really worked. It comes back to attackers innovating faster than defenders. And when technology emerges to protect devices more effectively, it takes years for new technologies to become pervasive enough to blunt the impact of attackers across a broad market.

The reactive nature of traditional malware defenses — in terms of finding attacks, profiling them, and developing signatures to block them on devices — makes the existing mitigations too little, too late. Attackers now randomly change what attacks look like using polymorphic malware, so looking for malware files cannot solve the problem. Additionally, attackers have new and increasingly sophisticated means to contact their command and control (C&C) systems and obscure data during exfiltration, making detection harder.

Attackers also do a lot more testing to make sure attacks work before they use them. Endpoint security technologies can be bought for a very small investment, so attackers refine their malware to ensure it works against a majority of defenses. This forces security professionals to look for different ways to break the kill chain, as we described in The CISO's Guide to Advanced Attackers[2]. You can do this a couple different ways:

1. **Impede Delivery:** If the attacker cannot deliver the attack to a vulnerable device, the chain is broken. For tactics like phishing this might be by blocking the email before it gets to an employee, or training employees not to click things that would result in malware delivery.

2. **Stop Compromise:** Even if the attack does reach a device, if it cannot execute and exploit the device the chain is broken. This involves a different approach to protecting endpoints, which will be the focus of this paper.

[2] https://securosis.com/research/publication/the-cisos-guide-to-advanced-attackers

3. **Block C&C:** If the device is compromised, but cannot contact the command and control infrastructure to receive instructions and updated attack code, the attack's impact is reduced. This requires the ability to analyze all outbound network traffic for C&C patterns, and to watch for contact with networks with bad reputations. We discussed many of these tactics in our Network-based Threat Intelligence research[3].

4. **Block Exfiltration:** The last line of defense is to stop the exfiltration of data from your environment. Whether via data leak prevention technology or some other content or egress filtering to detect protected content, if you stop data from leaving your environment there is no loss.

The earlier you can break the kill chain the better. But in the real world you need a multi-faceted approach, probably encompassing all the options listed above. Now let's dig into the Stop Compromise strategy, which is where application control fits into the security control hierarchy.

Stop Code Execution. Stop Malware.

The main focus of anti-virus and anti-malware technology has always been to stop malicious code from executing on a device, thus preventing compromise. The main areas of technological evolution involve how malware is detected and what parts of devices malicious software can access. We currently have a handful of approaches.

Default deny defines a set of authorized code that can execute on devices and blocks everything else. This provides true device lockdown because no code (either malicious or legitimate) can execute unless authorized.

1. **Block the Bad:** This is the traditional AV approach of matching malware signatures against code executing on the device. The problem is scale — there is so much bad stuff that you cannot possibly expect an endpoint to recognize every attack since the beginning of time.

2. **Improve Heuristics:** It is impossible to block all malware because it changes constantly, so you need to focus on what malware *does*, to the device or within the application. By improving recognition of attack patterns and blocking activity that makes no sense for a particular application, you can greatly improve your ability to protect devices from attack. Of course you need to know what the application *should* be doing at a very granular level so you can identify patterns that aren't "authorized application behavior" and likely indicate malware.

3. **Isolation:** An emerging technique for protecting endpoints is to run vulnerable applications — including browsers, Java, and Adobe Reader — in a restricted sandbox to isolate them from the rest of the device in case they execute malicious code. This assumes applications will be successfully compromised, but impedes attackers' ability to take over or steal anything from the device. An alternative is to descend into the innards of the operating system and isolate actual processes. This emerging technology is promising for making the base operating system resilient to attack.

4. **Allow the Good:** Finally let's cover default deny, which defines a set of authorized code that can execute on devices and blocks everything else. This provides true device lockdown because no code (either malicious or legitimate) can execute unless authorized. This approach underlies application control technology.

As you see, there are a number of ways to prevent compromise. The most appropriate approach depends on the specific situation. Before we get into use cases, let's dig into some perceptions of application control.

The Perception

There is no way to skirt this subject. Many organizations have had less than stellar experiences with application control technologies in the decade the technology has been available. The complaints tend to revolve around a few issues:

1. **Employees can't do their jobs:** This one is obvious. Employees are highly motivated to get work done, and disinterested in (security) slowing them down. Application Control defines a set of applications they can run and

[3] https://securosis.com/research/publication/network-based-threat-intelligence-searching-for-the-smoking-gun

blocks everything else, which can impede their ability to perform job functions. Some of the criticism is unwarranted — there may not be a business requirement to run iTunes or Gears of War on a corporate device. But the inability of a knowledge worker to install a new application when they need it is a legitimate concern. With the integration of more plug-ins and code execution within browsers, application control can also break the user experience of web browsing if it blocks plug-ins.

2. **It requires another agent:** This is another legitimate gripe. In order to enforce application control policies on a device, you need software agentry of some sort to run on it. This is no way around this. In the best case, the application control product can leverage another agent already on the device (for endpoint management, for instance).

3. **It's hard to manage:** In any organization of scale, employees want to do things and install code on their devices, all day, every day. Someone needs to approve or disapprove all these programs and determine their appropriateness. That takes time, and it's not like security folks have a ton of extra time for new responsibilities. Don't forget the need to authorize every patch or update for every application in use.

4. **It doesn't work:** This criticism is squishy, but given the examples of Microsoft's patching process being exploited, as well as malicious code running within authorized applications (such as browsers and Adobe Reader), it is possible to evade application control defenses. As with every other security control, nothing provides 100% security. Organizations need to understand the compromises involved in establishing their trust model for application control.

The Reality

These criticisms are all reasonable. Application control *does* impact user experience — it needs to. It's as simple as that. If employees can load arbitrary software onto their machines and execute code in their browsers and other applications, devices *will* be compromised. Every organization needs to weigh the trade-offs of security against usability to allow employees to do their jobs, balancing that risk against the productivity impact of locking down devices via security controls.

Approaches focusing on isolation or detection during (or after) compromise impact user experience less, but they depend on being right *every time* and catching *every attack*. They have historically proven unsuccessful. That doesn't mean none of the new isolation or advanced heuristics approaches holds promise. But at this point we do not know whether any new techniques can adequately address malware.

First you spend time profiling the main use models of devices and defining standard 'profiles', for which you can then design appropriate defenses.

Use Cases

Given the breadth of ways computing devices are used in a typical enterprise, trying to use a generic set of security controls for every device wouldn't make sense. So first you spend time profiling the main use models of devices and defining standard 'profiles', for which you can then design appropriate defenses. There are plenty of attributes you can use to define use cases, but we see a few in most environments:

1. **Operating System:** You protect Windows devices differently than Macs than Linux servers, because each has a different security model and different available controls. When deciding how to protect a device, operating system is a fundamental factor.

2. **Usage Model:** Next look at how the device is used. Is it a desktop, kiosk, server, laptop, or mobile device? We protect personal desktops differently than kiosks, even if the hardware and operating system are the same.

3. **Application variability:** Consider what kind of applications run on the device, as well as how often they change and are updated.

4. **Geographic distribution:** Where is the device located? Do you have dedicated IT and/or security staff there? What is the culture, and can you monitor and lock everything down? Some countries don't allow device monitoring, and some security controls require permission from government organizations, so those must be considerations as well.

5. **Access to sensitive data:** Do the users of these devices have access to sensitive and/or protected data? Depending on the sensitivity of the data, you may need to lock down the device. Contrast that with a public device in an open area, with no access to corporate networks, may be able to do with much looser or simpler security controls.

Using these types of attributes you should be able to define a handful of use cases or so, which you can use to determine the most appropriate means of protecting each device, trading off security against usability.

Let's consider a few key use cases where application control fits well.

OS Lockdown

We know you wonder why on Earth any organization serious about security — or even not so serious — would still use XP. That is a legitimate question with reasonable answers. For one, some legacy applications still only run on XP.

When an operating system is at the end of its life and no longer receiving security updates, it is a sitting duck. Attackers have free rein to continue finding exploitable defects with no fear of patches to cramp their style. Windows XP security updates officially end April 2014 — after that organizations still using XP are out of luck (as if *luck* has anything to do with it…).

We know you wonder why on Earth any organization serious about security — or even not so serious — would still use XP. That is a legitimate question with reasonable answers. For one, some legacy applications still only run on XP. It may not be worth the investment — or even possible, depending on legal/ownership issues — to migrate to a modern operating system, so on XP they stay. A similar situation arises with compliance requirements to have applications qualified by a government agency. We see this a lot in healthcare, where the OS cannot even be patched without going through a lengthy and painful qualification process. That doesn't happen, so on XP it stays. Despite Microsoft's best efforts, XP isn't going away any time soon.

Unfortunately that means XP will still be a common target for attackers, and organizations will have little choice but to protect vulnerable devices somehow. Locking them down may be one of the few viable options. In this situation using application control in *default deny* mode, allowing only authorized applications to run, works well.

Fixed Function Devices

Another use case we see frequently is fixed function devices, such as kiosks running embedded operating systems. Think ATM or payment station, where you never see the underlying operating system. These devices only run a select few applications built specifically for them. In this scenario there is no reason for any software besides authorized applications to run. Customers shouldn't be browsing the Internet on an ATM machine, so application control is appropriate on kiosks.

Similarly, some desktop computers in places like call centers and factory floors only run very stable and small sets of applications. Locking them down provides protection both from malware and employees loading unauthorized software or stealing data.

In both these use cases you will get little to no pushback from employees about their inability to install and run arbitrary software. Nothing in their job description indicates they should be loading software or accessing anything but the applications they need to do their jobs. In these scenarios application control is an excellent fit.

Servers

Another clear use case for application control is server devices. Servers tend to be dedicated to a handful of functions, so they can be locked down to those specific applications. Servers don't call the Help Desk to request access to iTunes, and admins can be expected to understand and navigate the validation process when they have a legitimate need for new software. Locking down servers can work very well — especially appealing because servers, as the repository of most sensitive data, are the ultimate target of most attacks.

General Purpose Devices

There has always been a desire to lock down general-purpose devices, which are among the most frequently compromised. Those employees keep clicking stuff, and are notoriously hard to control. Theoretically, if you could stop unauthorized code from running on these devices, you could protect employees from themselves. As we mentioned, end users push back against this because sometimes they legitimately need to install additional software. People get grumpy if they can't do their jobs.

Application control does have a role on general-purpose desktops — so long as there is sufficient flexibility for knowledge workers to load legitimate software. In most cases the application control software allows a grace period of a few hours to a day or so to run a new application, before it needs to be explicitly authorized by a manager or IT person. There are other situations where application control's trust model needs to be more flexible to meet the realities of enterprise use — such as permitting authorized software distribution products, authorized publishers, and trusted users to install & run software.

Of course loosening application control's trust model introduces a window of vulnerability for new malware to compromise devices. This enables employees to run new software to get their jobs done, but presents a tricky trade-off which requires careful balancing. We see many organizations deploying application control successfully this way, but be sure you have other controls in place — such as network security monitoring and malware callback detection — to identify compromised devices when application control isn't tight enough.

Application Control Selection Criteria

Now that you know the key use cases let's spend a little time highlighting some of the key features to look for in products.

1. **Library of executables:** If the application control product doesn't know about your applications it takes considerable work to get the system set up. A large and current application library is critical to both initial setup and scaling the technology, given the number of applications in use in a typical enterprise. You want a library which is kept updated, especially for patches and updates. Application control won't recognize updated versions of programs until they are explicitly added.

2. **Flexible trust model:** Speaking specifically of patches, you should be able to define certain software publishers whose code can run on your devices. That way, as long as code is properly signed by a trusted software vendor, it can run without manual authorization. Similarly, you should be able to define trusted distribution products that can automatically install software, trusted directories where applications can be found, file "owners" that ensure executables are installed by trusted accounts, and perhaps even authorize specific users to install their own software. Each decision to trust is a security trade-off, but they make the technology much more workable at scale.

3. **Policy setup:** The product should be able to monitor which applications are used on managed devices and build a baseline for the environment. This baseline can be used to quickly define which applications are legitimate and which are not for a good first cut at your policy base.

4. **Easy to manage policies:** The most resource-intensive aspect of deploying application control is keeping policies up to date, so you need an easy system for defining what is authorized and what is blocked for groups of users.

5. **Flexible enforcement:** You should have flexible options in case of policy violation. For instance you might want to allow an employee to run the software and alert the IT group. Or you may choose to allow users to run the application for a limited amount of time (a grace period) before it needs to be authorized. Or you could simply block execution. Policies should be based on device type, user, and/or group to satisfy your organization's security requirements.

Fortunately application control technology has been on the market for several years, with a number of offerings can provide all these capabilities. One other criterion to consider is leveraging other technologies already in place. For instance if you add application control as part of an endpoint protection or management suite, you can leverage agents already on devices to simplify management and policy maintenance.

Conclusion

Application control can be useful — particularly for stopping advanced attackers and securing unsupported operating systems. There are trade-offs as with any security control, but with proper planning and selection of which use cases to address, application control resists device compromise and protects enterprise data.

About the Analyst

Mike Rothman is an Analyst and the President of Securosis and author of The Pragmatic CSO <http://www.pragmaticcso.com>. Prior to joining Securosis he was a Vice President at META Group, and held senior positions at CipherTrust, TruSecure, eIQ Networks, SHYM Technology, and Ernst & Young .

About Securosis

Securosis, LLC is an independent research and analysis firm dedicated to thought leadership, objectivity, and transparency. Our analysts have all held executive level positions and are dedicated to providing high-value, pragmatic advisory services. For more information about Securosis, visit our website: <https://securosis.com/>.