Using Deep Instinct for Cyberthreat Prevention

Jake Williams
Deep Instinct is best described as an endpoint protection platform (EPP). Although it is not an endpoint detection and response (EDR) tool, it does provide some features that stray into the EDR space and definitely exceeds expectations for traditional EPP software. As we discuss in this paper, Deep Instinct takes a fundamentally different approach to detection than traditional EPP.

This paper details our review of Deep Instinct, a platform we found to be extremely capable at detecting and blocking threats. We explore the use of the system as well as features we discovered during our product review. We highlight use cases for features where applicable.

**Deep Instinct Architecture**

Deep Instinct uses a very lightweight agent that provides protection on the endpoint. Deep Instinct allows administrators to separate groups of installations into tenants. This separation is useful for managed security service providers (MSSPs) that want to use Deep Instinct for their customers, but it also can be useful for delegating authority over some portions of all installed systems. Different policies also can be applied to different tenants, maximizing flexibility.
Throughout the user interface, it is clear that the engineers behind Deep Instinct have actually done security work. As much as every vendor would prefer that you work only in their console, there are times when you need to perform data analysis that simply isn’t supported by their features. The option to export data (shown in Figure 1) from this tool in most interfaces shows the forethought applied to building Deep Instinct.

Some tools only support exporting data in CSV. As analysts know, this can cause problems with formatting. Deep Instinct exports data in Excel format, eliminating those issues.

Because Deep Instinct relies on a model rather than signatures for preventing threats (more on this later), it normally receives only a few updates per year. This schedule makes Deep Instinct ideal for protection of devices that may have limited connectivity (such as air-gapped assets on sensitive networks). In a more traditional antivirus model, detections are out of date almost as soon as the device is taken offline. But in the case of Deep Instinct, assets continue to be protected against unknown threats, thanks to the Deep Instinct detection model described in this paper.

**Deep Instinct Installation**

Installing the Deep Instinct agent couldn’t have been easier. Customers run the installer with a command line argument specifying the tenant ID the particular Deep Instinct installation should report to. One issue with installing new security agents is that there’s often a temporary post-installation performance effect. Some products inventory the state of the system to detect later changes. Others perform full filesystem sweeps to scan existing data. Full filesystem scans are not an issue for future deployments of new hardware, but they can cause a negative perception for existing end users.

Deep Instinct engineers have obviously had the “been there, done that, got the T-shirt” experience because the installer:

- Runs from the command line
- Doesn’t use complicated configuration files for unattended installation
- Installs with no confirmation prompts

Overall, the installation of Deep Instinct was straightforward and uncomplicated. One installation feature that was particularly interesting was the capability to install without performing a full scan of the filesystem for existing threats. Although security professionals would normally want to obtain an initial baseline scan on each endpoint, without this option we would find ourselves needing to schedule outage windows to accommodate the resources required during installation.
When it comes to the detection and prevention of malicious files, there are three primary methodologies:

- Detect the malicious file at the time it is written
- Detect the malicious file when it is opened/executed
- Detect the malicious behaviors during execution

Detecting a file when it is written is clearly the most ideal scenario. So why do many vendors focus on the time of execution?

Most endpoint protection tools on the market rely on runtime (behaviors) to provide many of the detections. Unfortunately, this approach creates a situation where a malicious file is able to be written to the target machine in the hopes that it will be detected later (at runtime). Perhaps even more unfortunately, this construct occurs by necessity. Signature-based detection of malicious files is extremely difficult due to the ease with which those signatures can be thwarted by simple obfuscation mechanisms.

Such a scenario leaves system owners in a position where they potentially have fissile material (dangerous files) on their systems while the endpoint protection vendor says, “Trust us, we’ll stop them from doing dangerous things if someone ever opens them.” This situation is analogous to a business owner buying more and more fire extinguishers rather than preventing flammable liquid from being poured all over the floor. The former is reactionary; the latter is proactive.

A final problem for traditional static analysis is that many endpoint protection vendors end up creating signatures that detect the obfuscation tool (packer) rather than the payload. Most packers write a new unpacking stub to the entry point of the executable file. Unlike the payload, this unpacking stub is relatively static in nature.

In the following sections, we review how Deep Instinct overcomes these challenges to deliver endpoint defense.
Artificial intelligence (AI) and machine learning (ML) seem to be the latest technology buzzwords. Every security vendor wants to brag about how its products use AI or ML. Putting aside marketing departments’ attempts to brandish the terminology, AI, ML and deep learning (DL) all refer to different forms of computer science technology.

Is AI/ML just a bunch of hype? Unfortunately, the answer is “It depends.” Many AI/ML algorithms require significant computational resources to run, and, as such, are probably not appropriate for real-time detection and prevention. AI/ML can, however, be used to train models that enable more accurate detection. That’s exactly what Deep Instinct does. By utilizing the DL algorithm (a specialized type of artificial neural network), Deep Instinct trains a model using its extensive computing resources. This model serves as what Deep Instinct refers to as the “brain” in its product, which is pushed to the endpoint for detection.

Deep Instinct’s model is significant for several reasons. First, it gives customers all the benefits of using AI at the endpoint without the overhead. Consider LASIK vision correction surgery—the technology behind LASIK was originally developed through research by NASA, but today we can get LASIK surgery very affordably in any number of locations, without an overnight hospital stay. Although NASA did the original heavy lifting work, we get all the benefits of enhanced vision very quickly and at a low cost. Deep Instinct is like NASA in this analogy: Deep Instinct does the heavy lifting, but organizations get all the benefit.

Second, and perhaps more importantly, Deep Instinct differs in the way it performs detection. A naive understanding of Deep Instinct’s model might suggest that it uses AI/ML to create signatures that are later used on the endpoint. Some vendors do this, but realistically speaking, AI/ML isn’t necessary in this signature-based approach. Rather than creating signatures, Deep Instinct is creating a model. (See the next section for more about what this means.)

Last, the Deep Instinct approach maximizes detection while minimizing false positives. We have probably all heard this assertion before, but in our product review, we found that both claims are actually true. Deep Instinct works because of its model vs. signatures approach.
How often have we heard the phrase, “Signatures are dead”? Probably more than a few times—and with good reason. Signatures are extremely fragile, and simply recompiling the same exact code can break an existing signature. While evading a signature isn’t always that easy, there’s no question that trying to keep up with emerging threats using signature-based detection is a losing game.

Deep Instinct uses DL to train models that are used to perform detection and prevention on the endpoint. The difference between using signatures and DL models is vast. In terms of evolution, comparing the two is like comparing the landline telephone with the smartphone. The technology used is fundamentally different.

Whereas signatures work to match file heuristics with malware heuristics, the neural networks of DL models detect malicious content by analyzing the entire file. If you compare neural networks to the human brain, they learn in a similar way. Take the toddler who is learning about dogs. The parent shows the child many different pictures of animals and many pictures of dogs. When the child later sees a dog, even if it’s a small dog that looks like a cat, the toddler is most likely able to correctly identify the animal as a dog. In a similar way, deep neural networks don’t just learn what is malicious, but they also learn the norm of what is benign. This capability to distinguish normal from malicious enables it to have far better detection accuracy, even for files it has never seen before.

Building Windows Policies

When bank robber Willie Sutton was arrested, he was asked, “Why do you rob banks when they’re the most heavily guarded?” Sutton famously replied, “Because that’s where the money is.” We use the same logic in this paper. We focus the majority of our time on Windows-related configuration and policies because that’s the largest attack surface in most enterprises. That said, Deep Instinct also supports agents, configuration and policies for:

- MacOS
- iOS
- Android
- ChromeOS

A killer feature of Deep Instinct is its ability to set differing levels of sensitivity for detection vs. prevention. Think of detection as the capability to log and generate output, whereas prevention actually blocks execution and generates a quarantine event. Every security engineer has considered turning up the protection level of their endpoint system at some point. But doing so comes with significant risk: While trying to enhance protection for confidentiality and integrity, security teams cannot jeopardize availability.

---

1 “Famous Cases and Criminals: Willie Sutton,” www.fbi.gov/history/famous-cases/willie-sutton
By separating detection and prevention sensitivity levels, Deep Instinct allows organizations to play out a “what-if” scenario by toggling the detection settings. Figure 2 is a screenshot of this separation. This functionality allows the organization to understand the impact before suffering the negative impacts that would come with just modifying the prevention bar.

Many organizations are struggling with potentially unwanted applications (PUA). These are applications that are often used for systems administration but are also useful for attackers. Deep Instinct supports granular permissions for PUA, as shown in Figure 3.

Deep Instinct also supports protection from threats opened via network folders. Although Deep Instinct doesn’t scan network file shares, anytime a user tries to open a file from one of these shares, it still provides protection. When vendors tell me about their file scanning capabilities, I always ask whether they scan network folders. Generally, I want the answer to be, “We don’t,” though I do like the option of enabling it in limited circumstances, because scans can cause all sorts of logging when the permissions to scan a file are missing, which doesn’t happen on a local machine due to system permissions. Additionally, if multiple endpoints running the software all target the file share simultaneously, performance issues are likely to result. Deep Instinct supports network file share scanning but doesn’t force it on users.

Deep Instinct also can protect against Dynamic Data Exchange (DDE), as shown in Figure 4. This is a form of active code that is critical to some MS Office workflows but is often used for malicious purposes. Although many endpoint protection solutions take an all-or-nothing approach to allowing DDE, Deep Instinct has more granular protection.
Behavioral Analysis

In addition to static analysis, Deep Instinct provides protection against threats at runtime as well by enabling behavioral analysis. Behavioral analysis options (shown in Figure 5) include protection for:

- Ransomware
- Code injection
- Execution of known payloads in memory
- Execution of suspicious and malicious scripts

Deep Instinct can easily target ransomware because it is implemented as a minifilter filesystem driver. A filesystem driver automatically sees every filesystem read, write and rename operation. Although ransomware itself can be implemented in any number of ways, the end result is the mass writing of encrypted file contents to files with common file extensions. With the right detection methodology (and Deep Instinct has it), stopping ransomware on a protected machine is a breeze.

Mitigating code injection is particularly problematic because it is also used for good. Most endpoint protection software uses code injection for process inspection. Many application plug-in architectures also use forms of code injection for just-in-time loading of infrequently used components. The trick isn’t detecting code injection—it’s differentiating good code injection from bad code injection.

Deep Instinct takes an approach to tackling code injection that obviously was created by someone who has been around the block a few times. Rather than simply blocking or allowing all code injection, Deep Instinct allows security administrators to log all code injection, giving them visibility without the fear of interrupting operations with false positives.

Upon initial inspection, we wondered whether even this wasn’t granular enough. There are, after all, some code injection operations this reviewer always wants to block (such as mimikatz injecting into lsass.exe). But there was no need to worry—Deep Instinct supports this by implementing the blocking of known payloads.

Figure 5. Behavioral Analysis
PowerShell Support

Most security products today support some monitoring and blocking of PowerShell code. It is often unclear, however, what specifically is being protected against. Deep Instinct separates detection vs. prevent actions for malicious vs. suspicious scripts (see Figure 6). These features use AMSI (built into Windows 10) for detection.

While blocking malicious scripts makes sense, many legitimate scripts today perform actions that are categorized as suspicious (for instance, changing the execution policy or executing PowerShell code in memory after downloading it). Detection and prevention include stopping in-memory attacks where code is executed without ever touching the disk. You definitely want a log of these, but realize they may be used in too many legitimate applications to block outright. Although event logs on Windows can capture this activity in the right configuration, there is no separation of suspicious vs. benign activity. Centralizing those logs from thousands of workstations is challenging at best.

Script Control

Granular protection against non-binary threats extends to PowerShell, HTA, ActiveScript (such as JavaScript and VBScript), and JavaScript executed under rundl132.exe. The latter is particularly interesting because it was used in a number of so-called fileless attacks. Unfortunately, simply blocking JavaScript execution through rundl132.exe isn't an option for many organizations because it is used in legacy workflows. Figure 7 is an example of this script control.
Allow-List Options

In any endpoint security product, false positives are the bane of a security administrator’s existence. Deep Instinct addresses the risk of false positives with extensive allow-list options. These are detailed as whitelisting options in the UI, but this paper will use the term allow-list. Deep Instinct supports creating allow-list options based on:

- File hash
- Script
- File certificate
- File path
- Behavioral analysis

Configuring allow-listing operations by file hash is an expected feature for most endpoint protection tools. Unfortunately, this type of configuration usually doesn’t solve the administrator’s issue with a false positive completely. They still have to hunt down the quarantined file and restore it for the user. Deep Instinct solves this issue by automatically restoring previously quarantined files that are now allow-listed.

Another critical feature of Deep Instinct allow-listing implementation is that it isn’t all or nothing. Too many security products support only global enabling of allow-listing. Not so with Deep Instinct. Any allow-listed entries in Deep Instinct can be applied to only a particular platform (Windows or MacOS) or to any configured policy. For an example use case where this would be desirable, consider RawCopy.exe, which is a tool for raw filesystem access used by forensics practitioners and attackers. In the vast majority of cases, we should probably block Rawcopy.exe, but if the tool is used on the machine of a forensics professional, we should probably allow execution. Deep Instinct supports this through extremely granular policy creation and application, an example of which is shown in Figure 8.

But that’s not the end of the story. Anyone who has managed allow- and block-lists knows that context is often lost. Sure, there’s a hash, but why is it here? What does it represent? Deep instinct solves this problem with the capability to add comments to any allow-list entries added. Although we would prefer not to be limited by character count, we typically use incident ticket numbers to provide full context so this is less of an issue. Deep Instinct also provides the file type for additional context so it’s easy to see whether a hash was an executable, zip file, or other such file.

Figure 8. File Hash Edit
Changing any allow-listing rule might create security issues. When investigating an incident, an analyst needs to know whether configuration changes were recently made. Deep Instinct supports this need by allowing analysts to filter based on the date a rule was added, as shown in Figure 9. One thing missing is the capability to see which analyst added the rule; but given the capability to filter by date, it’s only a small annoyance.

When evaluating a product, it’s always gratifying to spot evidence that the people designing it have actually done security work. When it’s clear that security personnel were involved in creating a tool, you can expect that future changes are likely to support operations rather than aligning with the “buzzword of the week.” Anyone who has done hash management for allow-lists knows what a pain it is to input hashes individually. Deep Instinct supports the option to import hashes by CSV and even supports adding comments in bulk for imported hashes. Figure 10 is an example of an allow-list import.

Although the option to import hashes via CSV is certainly useful in its own right, the fact that it exists at all is more significant. It indicates that real practitioners were involved in the product design—and that’s a gift that will likely keep on giving.

Behavioral analysis allow-listing supports granular permissions based on a given process, which means that if some legitimate processes must perform code injection due to their design (this is surprisingly common) they can be supported while preventing most code injection. Figure 11 is a screenshot of Deep Instinct’s behavior filtering. Although ransomware and known payload execution also are supported for allow-listing, these seem less likely to be used.
**File Hash Block-Listing**

Deep Instinct also supports block-listing by file hash. Just as with allow-listing, files can be added to the list by uploading a CSV. This feature allows organizations to support specific detections of known malicious files. Although it seems less likely that allow-listing will be needed, given Deep Instinct’s model-based detections, it is nevertheless a useful product feature. Even in the face of Deep Instinct’s formidable ability to detect malicious files, we would rather be able to confirm that our known-malicious hashes are being blocked.

Beyond the hypothetical use-cases, if Deep Instinct actually does miss a file that investigators decide is a threat, block-listing allows them to reconfigure Deep Instinct to provide immediate protection. Compare this function with the more traditional workflow of submitting a hash to an endpoint protection vendor and then waiting for a signature update. Clearly the Deep Instinct approach is superior.

**Exclusions vs. Allow-Listing**

In addition to allow-listing, Deep Instinct also supports exclusions. Exclusions are an interesting concept and promise to improve performance. Allow-list (and block-list) rules are only processed after Deep Instinct has completed its detection/analysis. By configuring exclusions, Deep Instinct doesn’t scan the excluded processes and folders. This approach offers potential hiding places for attackers if they understand the configuration of exclusions. As a result, we recommend that exclusions be used sparingly. However, they are a welcome feature to prevent performance issues, especially if there are directories or processes that are frequent fliers for false-positive events.

One fantastic use for folder exclusions are the quarantine folders used by other security products. Without this feature, if another security product quarantines a file first, Deep Instinct would detect it again, resulting in a second (and potentially confusing) alert.

**MacOS Policies**

Most organizations have at least some Macs in their portfolio. Unfortunately, most organizations struggle with protection plans for MacOS. While Macs are not immune from viruses, they are significantly more difficult for attackers to compromise than Windows machines in most enterprise configurations. Given this profile, how much should we care about protecting MacOS endpoints?

The answer is “A lot.” Most endpoint protection software views files coming from the internet with maximum suspicion (as is definitely warranted). But picture an attacker sending malicious files targeting Windows machines to a MacOS user. The MacOS user downloads the files and, because they aren’t malicious to MacOS, they are treated as benign. The MacOS user then shares the files internally. Because the files are now “internal,” they are treated with less scrutiny. This isn’t hypothetical; this author has worked multiple incidents where some variation of this scenario has occurred.
This outcome isn’t surprising. Most endpoint protection software relies on some combination of signatures and behavioral analysis to perform detection. But Windows files don’t execute on MacOS—even MS Office documents with malicious macros may not be detected. Signatures require working space in memory, so it makes sense to trim the signature working set to only those threats that directly target MacOS. Because Deep Instinct uses a model rather than signatures, the impact of scanning files on MacOS for threats targeting Windows is of little performance consequence.

Mobile Device Management (MDM)

Deep Instinct has options for protecting both iOS and Android devices. These options are significant because the perimeter is collapsing in most enterprises. In reality, it’s highly likely that users will access sensitive organization data via their phones at some point. Transfers of malicious files from mobile devices to desktops are a risk (and not a purely hypothetical one either). Protecting the entire environment is critical. Protection can’t stop at desktop devices and servers.

Running Deep Instinct on mobile devices acts as a mini-MDM, ensuring that organizations can detect jailbroken and rooted devices, both serious security risks. It also can detect SSL/TLS MitM attacks and alert administrators that they are occurring. Unsurprisingly, Android has more features than iOS, including detecting the installation of new certificates, USB debugging enabled, hosts file modification, and even MitM with ARP poisoning (not available on Android 10+ due to OS limitations).

Although Deep Instinct isn’t really aimed at the MDM market, the capability to gain insight into rooted and jailbroken devices may be enough for some organizations to check a compliance box, enhancing Deep Instinct’s value proposition.

Monitoring and Response

Although many users are likely to use a separate ticketing and alert tracking system, the monitoring features built into Deep Instinct are impressive. Analysts can monitor by:

- Events
- Devices
- Files

Events

Events are items that were either detected or blocked. The Event view offers significant filtering features. These include filtering by detection classification (shown in Figure 12), date, file type, device name, and searching by the details of the event (for example, file type).

![Figure 12. Classification Option Filters](image-url)
When an analyst drills down into event details, Deep Instinct immediately provides additional data that should aid in contextualizing the event. First, the Deep Classification engine reports what type of threat has been detected. As shown in Figure 13, details about the threat are then displayed, including:

- Whether the event was a detection or a block
- Whether the detection was from behavioral or static analysis
- The file type, path, and size
- File hash
- Whether the file is signed and, if so, certificate details

The file hash can be immediately searched in Google, VirusTotal, or AlienVault with a single click.

The Deep Instinct user interface also supports pivoting to other devices that may have seen the same threat. See Figure 14 for an example. This feature is extremely useful in prioritizing, particularly when a given threat has evaded detection (for example, a phishing payload that evaded detection at the email gateway).

The Process Chain and Life Cycle views provide detailed information about the threat (see Figure 15). This information is particularly useful when combined with an EDR (for instance, if the analyst wanted to terminate a process).
We especially like the *Logged in Users* display in the Event Details pane (see Figure 16). Although most products display the active user that generated the event, Deep Instinct goes a step further to show all users that are logged into the device at the time the alert was generated. This display is important because logged-in users may yield clues about the origin of the threat. Additionally, every logged-in user is at risk in some threats.

**Devices and Files**

The views for devices and files give security administrators other options for viewing and grouping detections. From the Details pane of a device, administrators can quickly pivot to detections seen on the device. Additionally, the *Devices* view Details pane (Figure 17) shows the status of the device the last time the device reported data to the Deep Instinct server.
Remediation Options

Deep Instinct supports multiple remediation options, as Figure 18 shows. The capability to right-click and add to an allow-list is a feature that every product should have—but many lack. This feature allows analysts to quickly address false-positive events. As a bonus, when a detected file is added to the allow-list, it is automatically restored to the machine from quarantine (if it was quarantined in the first place).

The response options also include the capability to delete a file remotely. This option is of particular interest when the event is detection-only but the administrator wants to prevent the file from being executed later.

Lastly, administrators have the option to upload the file to a D-Appliance, supporting additional static and dynamic analysis to be performed on the file. This provides better detections the next time a detection model (Deep Instinct calls this a “D-Brain”) is released.

One final remediation action not already discussed is isolating the host. Deep Instinct supports isolating a host to prevent a threat from spreading in the network. Although Deep Instinct is not an EDR, this is one of the response actions most commonly taken by EDR users.

Integrations

No product should be a silo, and Deep Instinct certainly is not. Although Deep Instinct supports connections to multiple SOAR (security orchestration and automated response) platforms via API, it’s always nice to see the old standby of syslog and email notifications. With these two tools at our disposal for this review, it’s easy to see that we can integrate quickly with practically any security environment. Figure 19 presents the remediation screen.
**Syslog Notifications**

All too often, security products offer only two options for **syslog**: nothing or everything. As shown in Figure 20, Deep Instinct offers granular choices for what types of events are forwarded to the **syslog** server. These options are ideal for organizations that have to use centralized alerting but will log into the console to investigate the event in detail.

**Email Notifications**

The same set of notifications configurable for **syslog** is also available for email notifications, which allows security administrators to be granular in their alerting, sending different information via **syslog** and email alerting.

**API Usage**

No tool being deployed today should be without API integrations. APIs allow the easy use of SOAR platforms, in turn increasing analyst efficiency. Deep Instinct supports API keys for read-only or read and remediation activities that can be assigned at the individual tenant level (see Figure 21 for details). A single API key also can provide access to multiple (or all) tenants. We did not evaluate API use as part of this product review, but it’s good to know it is available.
Like any modern security product, Deep Instinct supports reporting directly from the console (see Figure 22). The reporting is configurable for different periods and can be scheduled to automatically email the report when SMTP is configured.

Reports include exactly the sort of summary data expected, including the number of files scanned and the number of detections. Figure 23 provides a sample report. It also includes a summary of remediation actions and false positives. Although most security analysts won’t use these reports on a daily basis, they are incredibly helpful in generating metrics when management asks.
Reporting also covers the types of threats detected, separated by file type and threat classification, as shown in Figure 24. This coverage can help guide decisions about where additional defense in depth may be warranted, particularly when Deep Instinct is the only tool that detects the threat.

One of our favorite features of the report is the legend of threat classifications (see Figure 25). Too often, security products generate reports they claim are intended to be consumed by executives but make extensive use of technojargon. Deep Instinct solves this problem by including a legend in the report.

![Figure 24. Malware Demographics](image)

![Figure 25. Report Legend](image)
As shown in Figure 26, reporting also separates threats by the type of system, which can be extremely useful in determining where additional coverage may be needed, particularly in less popular device types (such as those using the Chrome OS, for instance).

The report also includes a risk summary, as illustrated in Figure 27. The numbers are shown in aggregate and then broken down by device and user. This breakdown is extremely helpful when a single device or user has significant numbers of alerts that are skewing the overall totals.

Conclusion

In this paper, we reviewed Deep Instinct, an endpoint protection platform that offers significant protection from threats using models trained by artificial intelligence. We found the product provides excellent protection and simultaneously solves a number of visibility issues normally only addressed by far more complex solutions. Deep Instinct also has a small enough footprint that organizations can run it in parallel with other products, such as EDR, providing defense in depth. Those organizations in the market for a turnkey endpoint protection platform should consider deploying Deep Instinct in their network.
About the Author

Jake Williams is a SANS analyst, senior SANS instructor, course author and designer of several NetWars challenges for use in SANS’ popular, “gamified” information security training suite. Jake spent more than a decade in information security roles at several government agencies, developing specialties in offensive forensics, malware development and digital counterespionage. Jake is the founder of Rendition InfoSec, which provides penetration testing, digital forensics and incident response, expertise in cloud data exfiltration, and the tools and guidance to secure client data against sophisticated, persistent attacks on-premises and in the cloud.

Sponsor

SANS would like to thank this paper’s sponsor:

Deep Instinct