

Why Traditional EDR Is Not Working—and What to Do About It

Written by **Jake Williams**

June 2019

Sponsored by:

McAfee

Introduction

If you work in infosec and haven't been living under a rock, you have undoubtedly heard of EDR. EDR, or endpoint detection and response, promises to revolutionize the way security analysts neutralize attacks. Unfortunately, like many other solutions in the infosec space, EDR has fallen short of the promised hype.

A typical sales pitch for an EDR goes something like this: "Sure, we all know you can check for alerts in the SIEM, open an escalation ticket in your productivity software suite, and alert a systems administrator to take action. But in the time that all takes, the attacker may well have performed a smash and grab, thus stealing your data. The problem isn't that you don't have enough detection capability; the problem is that it's all over the place. Even then, how are you going to respond? Time is money, and time favors the attacker. That's why you need EDR. It consolidates the detection functions and the response functions into one platform."

But as you've probably guessed, EDR has not quite lived up to the lofty expectations.

One of the largest failings of EDR deployments is that the EDR simply doesn't integrate with the other tools (SIEM, IDS, DLP, etc.) used by security analysts. Another common issue with EDR deployments is the typical separation of duties between IT and cybersecurity. The EDR platform crosses both lines of business; cybersecurity traditionally performs the detection function, while IT traditionally performs the response function. Any time a response function is performed, there is some risk of a system outage. Because IT will take the blame for any system downtime, it only makes

sense that they have traditionally managed all response functions. And because EDR covers traditional detection and response functions, it breaks that paradigm, paving the way for process and workflow issues.

Even if there are process issues to deal with in the organization, many EDR platforms themselves are unable to function as the true one-stop solution promised by the vendor. Many EDR platforms lack integration with a SIEM. Others focus entirely on the endpoint, completely missing the role that network traffic plays in both eliminating false positives and providing valuable context for true positive alarms.

In the following sections, we examine other challenges with traditional EDR platforms and what you can do to overcome those challenges for effective EDR implementation. We also introduce a checklist of things you should consider when selecting and deploying an EDR platform.

Root Causes of EDR Shortcomings

Like many infosec systems, EDR has a history of failing to live up to its advertised capabilities. Similar to SIEM deployment disappointments, this performance would seem counterintuitive. Given that the products are relatively mature, there must be a common thread among their failures. In this section and in Figure 1, we examine several reasons why EDR systems often fail to live up to advertised claims when deployed in the field.

Failure to Integrate Data from Other Sources

The first, and perhaps most common, root cause for EDR system deployment failure is that the EDR is not integrated with data from other sources. While the most commonly requested integration with the EDR platform is the SIEM, numerous other data sources can help enrich alerts from the EDR as well as eliminate false positive alarms. Ideally, EDR tools would integrate with the following:

- DNS logs
- Network traffic flow logs or NetFlow
- Web proxy logs
- IT inventory data
- Vulnerability scan results
- DHCP logs
- Active directory log on information
- MFA provider logs

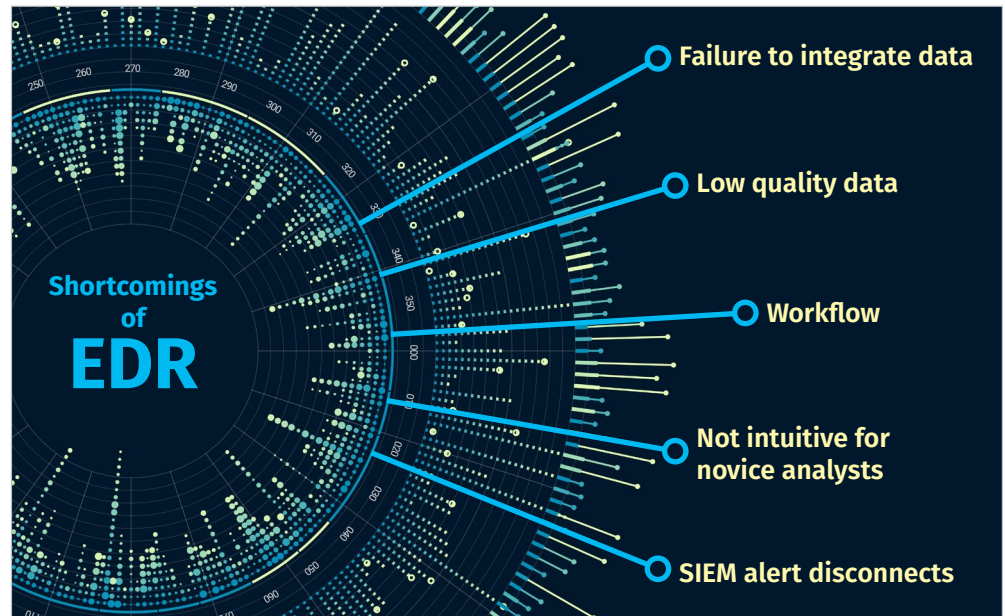


Figure 1. EDR Shortcomings

Although the list above is certainly not all-inclusive, most of today's EDR platforms do not integrate with these data sources. Some EDR vendors will note that most data types listed above can be ingested into the SIEM system and that, in turn, their EDR platform integrates with the SIEM.

While this may be true, why would we want to use the SIEM through the EDR if that is the only integration offered? Why not then simply forward EDR data to the SIEM and analyze the data there? That is an option, but it treats the EDR as an ED (a detection tool with no response option). After we have decided on a course of action, we must change context from the SIEM to the EDR to perform the response action. This clunky workflow fails to embody the promises of efficiency that drove the adoption of an EDR solution in the first place. It also positions the EDR as subordinate to the SIEM.

Low Quality Investigation Data

Low quality investigative data is another common shortcoming of EDR. Practically every EDR solution on the market today can examine endpoint data such as process lists, registry keys and values, and active network connections. This data is valuable in identifying an intrusion, but that data itself is incomplete. It lacks the context that other tools such as a SIEM would use to illuminate false positives and highlight true positive alerts.

An example of low quality EDR data causing a missed alarm is that of password spraying. In a password spraying attack, the attacker attempts to use the same password across multiple accounts in the domain (often all of them). Without integrating logs stored in the SIEM, the EDR may miss this attack.

Low quality data is not a critique of any particular EDR system (or even EDR systems in general). Rather, it is more of a visibility issue. EDR systems typically analyze data from a single endpoint at a time, while a SIEM correlates higher level data across multiple endpoints (including network data). Many organizations expect an EDR to replace a SIEM for detections on the endpoint but are disappointed to discover they aren't getting the full picture from the EDR.

Too Many Pivots Required to Complete Quality Analysis

In many cases where incomplete investigations are performed, all the data available to complete an investigation is actually available to the analyst. The core problem is that the analyst fails to realize the data is available because they are looking in the wrong place. This often happens when the EDR system lacks integration with the SIEM. The analyst receives an alert in the EDR system and then must pivot into the SIEM to get additional data. Unfortunately, the data may not be stored in the SIEM in a manner that facilitates easy investigation.

A common cause for this issue is DHCP. Many logs in the SIEM will only contain IP address information. However, other logs will contain only hostname information. Of course, some logs contain both. Most EDR systems use some sort of installed agent on the endpoint, so the installation ID makes the most sense as a unique identifier for the

endpoint. However, this information is unlikely to be captured in the SIEM, which means that multiple pivots must be performed to correlate and alert between the EDR system and the SIEM.

Things become further complicated if all the requisite data is not available in the SIEM. For instance, DHCP logs (critical for correlating other valuable data) are often not populated in the SIEM at all. In other cases, DHCP logs are forwarded to the SIEM but have a shorter retention than other data that requires the DHCP logs for context. Each pivot required to give the analyst a complete picture of the event reduces the likelihood of a comprehensive investigation.

Novice Analysts Often Struggle to Hit the Ground Running

Many EDR systems fail to live up to their promises in real world applications because they offer the analyst too many options. This overabundance of configuration options can, and often does, overwhelm less experienced analysts.

Since EDR found its place in the overall infosec ecosystem, it became generally understood to have an active role in querying endpoints. Meanwhile, a SIEM had a passive role (ingesting and correlating log data). EDR's active role, however, comes at a cost. Making full use of the EDR requires knowing what questions to ask, a concern that disproportionately impacts novice analysts.

You can ask questions of the SIEM, but your options are limited to the data already stored. This limitation leads to some difficult questions that engineers must answer before an incident, as shown in Figure 2.

Although we can rely on best practices to answer some of these questions, it is impossible to accurately answer all of them for any specific incident within a specific organization.

The EDR system fills this void. The SIEM stores and correlates the logs most likely to be useful during an investigation, but the EDR system gives the analyst maximum flexibility to ask questions they didn't know they would need to answer. One notable place where EDR fills the gap is when new classes of attacks emerge. Because threat actors are constantly adapting their techniques, organizations need to be able to just as quickly adapt their investigative techniques. This is where EDR outperforms other technologies on the market today: with the flexibility to actively query protected endpoints.



Figure 2. Logging Architecture Considerations

EDR can help solve the data hoarding problem some organizations experience with other solutions as well. Because EDR allows the analyst to query endpoint data in real time, there is less of a need to “log everything just in case we need it later.” When the need for particular endpoint data becomes apparent, analysts can satisfy the data collection requirement.

Suppose a new adversary is discovered using a new DLL sideloading attack in the organization’s environment. To investigate the attack, the analyst needs to discover the loaded DLLs on every endpoint in the enterprise. While this data could technically be logged every time a new process is launched and reported to the SIEM, the size of the data makes this approach impractical. A far better approach is to query this data on-demand using the EDR to find all processes where the malicious DLL has been sideloaded into victim processes. Not only does this require less storage, but the query is also likely to complete faster with real-time data from the EDR than with historical data from the SIEM (which may no longer be applicable anyway).

Vastly different skill levels are required to process an alert from the SIEM than to ask unstructured questions of the EDR. Novice analysts struggle with not knowing what questions they need to ask in the first place. As a result, they often report lower perceived EDR value. This situation is compounded in organizations that lack a heavy bench of senior infosec analysts. EDR solutions that provide intuitive interfaces and investigation playbook creation tools help close the gap for novice analysts, enabling them to hit the ground running.

SIEM Alerts Are Disconnected from Alerts in the EDR

In most EDR deployments we have observed in the field, there is a disconnect between alerts generated by the SIEM and the EDR. In some cases, this is due to a lack of integration, and in other cases it is due to processes that were put in place before the EDR was deployed. The adage “you can’t teach an old dog new tricks” certainly applies to the latter case.

But if the EDR can easily ingest SIEM data and use that to either generate alerts or add context to existing EDR alerts, then perhaps the old adage doesn’t apply after all. The simple fact is: EDR systems that can’t ingest data from other sources are creating another silo solution. Given the rapidly changing face of today’s attacks, organizations can’t afford to deploy solutions that don’t readily integrate with others.

Many SIEM products integrate a ticketing system within the software itself. Many organizations use the SIEM integrated ticketing solution as their case management and alert tracking tool. Unfortunately, most EDR systems cannot simply integrate with these proprietary case management tools.

One way to avoid this issue is by future proofing the security operations architecture. When possible, avoid proprietary solutions without external interfaces. While there are additional licensing requirements and configuration man-hours required to use an external ticketing system, it is probably worth the upfront investment to facilitate future architectural changes.

Even analysts who have experience using a SIEM may find the active nature of an EDR less intuitive. EDR platforms that support user definable workflows offer better value for organizations with novice analysts.

When a centralized ticketing system is used, both the SIEM and the EDR feed into the same system. This creates a single location where the analyst can identify all relevant alerts, regardless of whether the system generated the alert. Unfortunately, in far too many cases, the analyst receives the alerts from the SIEM immediately as part of the established workflow, while they must regularly poll the EDR system to see if any new alerts are present. A stopgap solution to this problem can be to feed EDR alerts to the SIEM and then configure correlation rules on the SIEM to automatically generate tickets for EDR alerts. This solution is far from ideal, however, as the EDR is effectively rendered a second-class product to the SIEM.

EDR—Not Just About Detection

Thus far, this paper has focused on the detection capabilities and workflows of an EDR system. But an EDR is about more than just detection. Clearly response capabilities are important as well. When evaluating EDR systems, however, users often discover that the remediation and response capabilities are bolted on as an afterthought to the detection capabilities. As we examine product lifecycles, we also commonly observe that detection receives far more feature enhancements than the response counterparts.

We have already highlighted one reason for this: an EDR provides capabilities that have traditionally been part of the cybersecurity as well as the IT team work roles. However, despite straddling capabilities employed by two different teams, the funding requirements for EDR systems are most often borne by cybersecurity departments alone.

It would appear EDR vendors are by and large responding to market forces. Vendors assume decision-makers in the cybersecurity teams are much more likely to value detection features than they are response features. We believe, however, that this assumption is wrong. Many of the query functions that can be performed by an EDR can also be performed by traditional IT asset management software, such as SCCM. The major benefit of deploying EDR only becomes apparent when the response functions of EDR are factored in. In this section, we highlight some of the ideal response features for EDR systems.

Ideal Response Features for EDR Systems

- Terminate running processes
- Prevent processes from executing based on name, path, arguments, parent, publisher, or hash
- Block specific processes from communicating on the network
- Block processes from communicating with specific hostnames or IP addresses
- Uninstall services
- Edit registry keys and values
- Shut down or reboot an endpoint
- Log users off an endpoint
- Delete files and directories from the operating system, even if they are actively in use (may require a reboot)

Most EDR systems offer some subset of these capabilities. One of the most frequently overlooked is the machine restart. Malware authors commonly install malware that employs user mode rootkit hooking. Hooks that prevent the deletion of registry keys and values used to persist the malware between reboots are installed in the operating system. Files and directories used by the malware are hidden from view by processes while the malware is running. The malware is often persisted via a service that is regulated by the service control manager.

Without the capability to perform a remote reboot on a system, malware that employs the techniques listed above may be impossible to remediate using EDR. Analysts need the capability to quickly react to and contain an incident. The old school approaches of simply reimaging a system every time there's an antivirus alert simply don't scale to today's threats.

Most EDR systems today allow the analysts to terminate individual processes. However, many of those same systems lack the capability to prevent an identical process from restarting immediately. They literally leave the analyst playing a futile game of whack-a-mole where the attacker always has the advantage and the defender is forever trying to catch up. When evaluating an EDR system, carefully examine how the EDR system allows analysts to proactively prevent a known attacker technique rather than simply respond to it.

Another common problem with EDR systems is the use of script interpreters such as PowerShell or cscript. Both script interpreters are used regularly for normal system operations, but they are both frequently used by malware as well. The EDR system must afford the analyst some capability to differentiate between legitimate and illegitimate use of the script interpreters.

Checklist for Evaluating EDR Solutions

During the past several years, the EDR solution space has become relatively crowded. Organizations must properly evaluate an EDR solution if they want to maximize the probability that the EDR will meet their mission needs. This section provides a checklist for features that will ideally contribute to EDR deployment success in an organization. Not every feature mentioned in the checklist is necessary to ensure success in every deployment, but each is included because it has been shown to be an important capability to an organization.

Feature	Priority	Justification
Capable of querying all endpoints, pre-configured groups of endpoints, or ad-hoc groups created by other query data	HIGH	During an investigation, analysts will need to configure ad-hoc groups that cannot be anticipated at EDR installation time. Often, groups will be created based on another query performed against the EDR (for example, query all hosts where userX has logged in within the prior seven days).
Response functionality meets the anticipated needs of the organization	HIGH	The organization should review incidents that have been worked in the past and examine the response actions taken manually to determine whether the EDR can replace them. An EDR that does not offer response options appropriate to the organization workflow should be given lower weight.
SIEM integration	HIGH	SIEM integration is a highly desirable feature of an EDR. Ideally, integration should be bidirectional, but the EDR must be able to feed data to the SIEM.
Workflow support for novice analysts	HIGH	While providing maximum flexibility to senior analysts, the EDR should support predefined (and configurable) workflows for less experienced personnel who require more guidance during an investigation.
Ticketing system integration	MEDIUM	The EDR should be able to feed data to a third party ticketing system (for example, JIRA).
Separate privilege groups for query and response actions	MEDIUM	Separating privileges for creating queries and performing response actions is critical to gaining adoption of the EDR platform. Many users who need to query data (for example, threat hunting) should not have the capability to terminate processes such as, deleting files.
IOC processing	MEDIUM	Ideally, the EDR can process IOCs in multiple formats, such as Yara and OpenIOC, but at least one format should be supported.
Hypervisor integration for scan groups	MEDIUM	In modern IT environments, servers migrate seamlessly between physical hypervisors. EDR scans consumer resources on guest servers, which can cause a hypervisor to be oversubscribed. If the EDR supports hypervisor integration, scan groups can be dynamic.
Syslog integration	LOW	The EDR should be able to send alarms via syslog to provide maximum flexibility for integration with other systems.
Scan impact throttling	LOW	The EDR should offer configuration options to limit the impact of scans being run. Business interruption is one of the issues cited by users that limits adoption by organizations.
Multiple data export locations	LOW	Some products allow integration with only a single follow-on system (syslog can only have a single destination, for example). This limits flexibility when building a complete system architecture.
API integration	LOW	API integration allows the organization to customize data enrichment and response activities. Organizations that automate the interactions between systems today may reprioritize this item.

EDR Use Cases

The obvious use case for an EDR system is in detecting intrusions and automating the response. In this section, we walk through some use case examples of how an EDR might be operationalized in the field. There are certainly more use cases than we can cover in this section, but these examples can highlight the types of workflows enabled by implementing an EDR solution.

The EDR Detects a Process Named lsass.exe Running as a Regular User

The Situation

The process named **lsass.exe** should only ever execute under a system context, never under a normal user context. Because a novice analyst might not know that only a single instance of the **lsass.exe** process should be running on a Windows host, the EDR highlights that fact and triggers an alert, starting an investigation. In processing the alert, the analyst queries the EDR for running processes on the endpoint and discovers that a rogue **lsass.exe** process is running as a child of **cmd.exe**. The **cmd.exe** process is running as a child of **winword.exe**. The analyst now suspects that the rogue process is likely the result of a phishing attack with a malicious Word document. The start times on the processes lead the analyst to conclude that the phishing email was just opened, presenting an ideal opportunity for the EDR to take the incident from intrusion to detection to remediation in a span of only minutes.

EDR's Role

Recognizing that the rogue **lsass.exe** process is certainly malicious, the analyst queries for network connections involving any of the rogue processes and finds that the **lsass.exe** process is communicating with an IP address in eastern Europe. The analyst uses the EDR to terminate the rogue processes (**lsass.exe**, **cmd.exe**, and **winword.exe**) immediately. The analyst also uses the EDR system to query for any endpoints that are communicating with the suspicious IP address. The analyst discovers two additional endpoints and requests system information (running processes and service configuration information, for example) from those systems. These systems aren't using a rogue **lsass.exe** process like the original, but the EDR makes it easy to identify the malicious processes anyway.

Traditional network monitoring systems (such as NetFlow and full packet capture) lack the granularity that EDR provides. While traditional network monitoring systems can point the investigator to an endpoint that is communicating with a suspect IP address, they stop short of identifying the actual processes involved. This aspect of EDR unfortunately means the investigator must involve yet another system in the investigation. On the other hand, the EDR allows the investigator to identify the specific processes involved in the communication (and terminate them).

Although the first system has just been compromised and can be easily dealt with, it is not yet known when the newly identified machines were compromised. The analyst dives into the information returned from the EDR and discovers the persistence mechanism as being a LNK file in the user's startup directory, which is removed using the EDR.

The analyst uses the EDR to collect other forensic artifacts on the system, including open file handles from the malicious processes identified. This action helps identify a previously unknown file used by the malware, which is subsequently passed to the organization's reverse engineers. Reverse engineers are able to decode the file, which contains a list of callback domains used by the malware. It's important to note that this file (and the newly discovered IOCs) would not have been found without the help of EDR.

Despite having different process names and file hashes, the malware identified that the three systems all share the file version number "1.0.29.5." The analyst uses the EDR to query for any systems running processes with this executable metadata. This sort of detection simply isn't possible without EDR because this level of detail is something that would never be logged with a traditional SIEM.

To close out the response, the EDR is used to terminate any remaining running processes, confirm the removal of the **.LNK** file used for persistence, and to block network communication with the discovered IP addresses and domain names.

This case study highlights the value of the EDR for multiple reasons. First, on the initially detected machine, the incident is detected and remediated from a single pane of glass in minutes. Second, the analyst is able to immediately identify the processes communicating with the malicious IP addresses in real time (rather than pivoting between systems). Finally, the analyst can perform queries for information that would never be logged in a SIEM (the file version number, for instance), ensuring that no variants of the malware remain undetected in the environment.

EDR Detects Suspicious Behavior

The Situation

The organization receives new threat intelligence that an APT threat known to target them is using the directory **%USERPROFILE%\AppData\Roaming\SharePoints** to stage data for exfiltration. The organization has not observed activity from the APT group in its network recently and is concerned that the newly released indicators might be observed in their network. If indicators are discovered, the organization wants to be able to react quickly and remove the attacker from the network.

EDR's Role

The EDR is tasked to query each machine on the network for the presence of the directory **%USERPROFILE%\AppData\Roaming\SharePoints**, which was identified through threat intelligence as an indicator of compromise. The directory is discovered on four machines, two at headquarters and two in different remote offices where the organization does not have on-site IT or information security staff. Detecting an intrusion is always less than ideal, but detecting one where there's no onsite support creates additional complications.

The analyst immediately queries process information, including loaded DLLs and network connection data from the four affected machines through the EDR. The analyst doesn't immediately see any processes that appear malicious but does observe an unfamiliar DLL, **kernel164.dll**, loaded into **explorer.exe**'s (the user's desktop) address space. Finding the DLL loaded into **explorer.exe** is consistent with the shared IOC, which is also tied to the user (rather than the machine).

Examining the network connection data, the analyst notes that on three of the infected machines, the **explorer.exe** process has a connection to an external IP address on TCP port 33389. The analyst considers using the EDR to immediately block communication with the IP address but decides against it until additional queries can be made of the EDR.

Using the knowledge gained from the previous queries, the analyst uses the EDR to query for all machines communicating with TCP port 33389 or the suspect IP address. He also queries for all processes loading a DLL name **kernel164.dll**. The analyst finds five new machines that are loading **kernel164.dll**. Because these machines do not have the staging directory, they would not have been located using the original threat intelligence provided. The analyst also discovers another suspect IP address.

The analyst uses the EDR to query the identified computers for the registry settings of all logged-in users to discover the persistence mechanism used by the malware. All infected machines have an autorun entry to start a legitimate third-party system profiler application. Because the application is digitally signed, it was allowed through application whitelisting but is being used to sideload a DLL from disk, inject into **explorer.exe**, and exit. Additional EDR queries are performed to look for other machines with the third-party system profiler installed, but none are discovered.

With detection complete, the analysts now transition to response. They use the EDR to remotely remove the autorun registry keys used for persistence and the third-party system profiler (and the sideloaded malicious DLLs). They also use the EDR to block all communication with the identified IP addresses. While communication should also be blocked at the firewall, this task is typically under the management of another team. Coordinating across multiple teams during response leads to delays. Although it may not have been appropriate to block communication with the IP addresses during the identification and scoping phases, blocking should be immediate and seamless when the incident responders pull the trigger. In our experience, that coordination with the network team is neither immediate nor seamless, further highlighting the value proposition of the EDR.

The analysts also use the EDR to collect the files in the staging directories to assess the impact of the incident. After collecting the files, the analysts use the EDR to remove the staging directories (and all files collected by the attacker) from the infected machines. As a final step, the machines are rebooted remotely using the EDR (a reboot is the best practice due to the code injection technique used) and the machines are re-scanned after they are logged in to confirm they are clean.

The value of the EDR in this incident is particularly high in locations that have no dedicated IT or infosec staff. Without the EDR, the organization could not have responded to all identified machines simultaneously across multiple sites. In an age where attackers are becoming ever more sophisticated, organizations must ensure they act swiftly, denying the attacker the opportunity to counter their response by deploying new, unknown malware.

Conclusion

The EDR market, once a niche market, has exploded in recent years. Just because the product carries an EDR description, though, doesn't mean those products have anything approaching feature parity with one another or have been proven particularly effective. In this paper, we discussed features to look for when considering an EDR, use cases for deploying an EDR, pitfalls in current EDR deployments, and a checklist for evaluating EDR products. Organizations considering an EDR product should consider the advice presented in this paper when evaluating their own deployments, including:

- Using the EDR features checklist
- Considering lessons learned to avoid EDR deployment issues
- Ensuring the chosen EDR solution implements the appropriate response features
- Determining up front how integration with SIEM and other tools will impact EDR deployments
- Ensuring the EDR supports workflows that are usable by novice analysts

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 attack on-premises and in the cloud.

Sponsor

SANS would like to thank this paper's sponsor:

