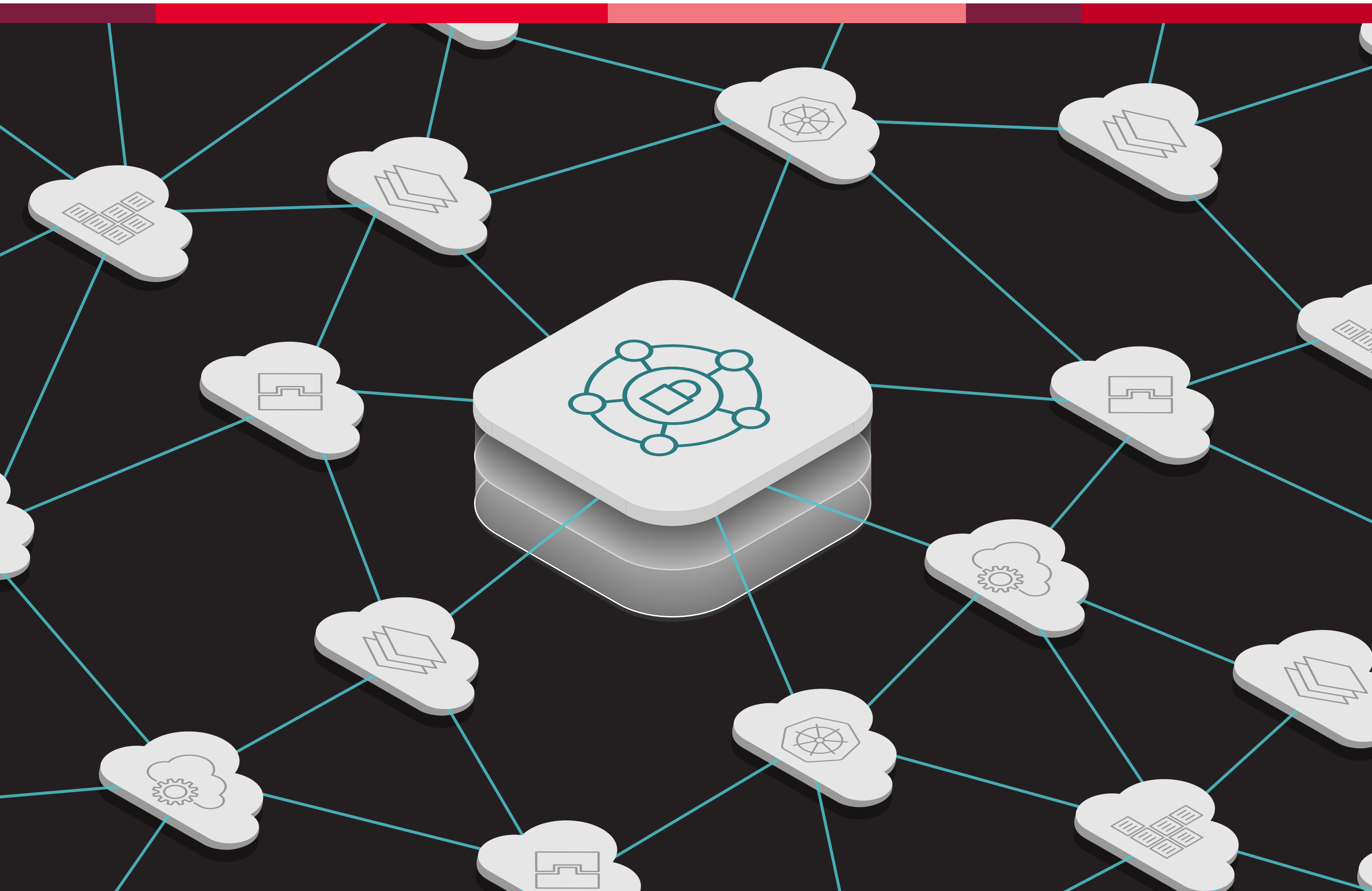
WHITE PAPER

A Closer Look at the F5 Distributed Cloud AIP Agent

A white paper on our Agent's components and data collection, and how they work to protect application infrastructure.



F5® Distributed Cloud App Infrastructure Protection (AIP), formerly known as Threat Stack, consists of a number of components, but the piece that directly interfaces with your host operating systems is the **Distributed Cloud AIP Agent**.

Before deploying the Agent to your production environment—or even before considering a trial in a development environment—it's natural to have questions about what this piece of software is doing and how it works. This paper unpacks these questions by looking at the Agent's components, supported systems, types of data collected, and some brief use cases for applying it.

Overview

The Agent is implemented in Go and C, and it collects data on low-level operating system activity for Linux or Windows Server systems, file integrity monitoring (FIM), and user space events. It does not sniff network traffic or perform packet inspection. The host-OS-centric view of the Agent does, however, provide valuable context about the process-level metadata surrounding a network connection, observing which commands that a particular user ran prior to opening it, as well as ports, IP addresses, and related arguments passed at runtime.

Linux Roots

Distributed Cloud AIP was originally designed to piggyback off existing Linux subsystems, and to offload the complexity of gathering logs and surfacing anomalous behaviors. While the Agent itself consists of several data collection components, the core piece is `tsauditd`, the replacement for the Linux Audit daemon `auditd`.

For those unfamiliar with the Linux Audit system, it consists of two main components: a set of user space processes and a component for "kernel-side system call processing."[1] One of these user space pieces is `auditd`. So `tsauditd` plugs into the existing Linux Audit system from user space to log kernel system calls—without having to load an additional Linux security module (LSM). In fact, all the pieces of the Agent for Linux run in user space.

The Agent for Linux

The Distributed Cloud AIP Agent for Linux consists of four processes. Some of them can be disabled depending on the environment, but typically, a combination of these will be running:

- `tsagentd`: worker process spawned by systemd to manage inputs from various sensors
- `tsauditd`: our replacement for auditd that consumes, processes, and transforms raw kernel audit events with high-performance, low-latency, and minimal-compute resource usage—also performs targeted FIM via built-in filesystem APIs (inotify and fanotify)
- `tscontainerd`: sensor to gather events from Docker containers

- `tskubes`: sensor that filters data from the Kubernetes Events API for orchestration events and other security-related metrics

Deployment Options

The Agent, in production with customers since 2014, has several deployment options that can be mixed and matched to suit your infrastructure and operations workflows:

- Install the Agent on Linux hosts manually via `apt` or `yum`
- Bake it into machine images, then pass in deployment keys via a startup script
- Automate installation using Chef, Puppet, Salt, or Ansible
- Deploy the Agent as its own Docker container, and optionally include it in Kubernetes DaemonSets. While this version of the Agent runs as its own container, it runs once per worker node and maintains the same in-depth visibility and functionality as host-based Distributed Cloud AIP Agents

New eBPF Capabilities

The Distributed Cloud AIP Agent for Linux now includes a new eBPF component that reports additional network and DNS telemetry. eBPF safely and efficiently extends the Linux kernel's capabilities without requiring you to change the kernel's existing code or modules and allows you to add additional capabilities to the operating system (OS) at runtime. eBPF enables event-driven custom code to run natively in your OS kernel without requiring changes to applications or the kernel to observe and enforce runtime security and observability.

Our Agent for Windows Server

The Agent for Windows Server is purpose built to efficiently collect security signals from native Windows subsystems, and to run its own proprietary minifilter driver for file integrity monitoring. While core Windows Agent functionality is written in Go, the FIM minifilter driver is implemented in C++. The FIM driver is signed by Microsoft.

The Windows Agent surfaces security events from the Windows Event Log. If Sysmon runs on your servers, it can be configured to capture Sysmon events to add metadata about "process creations, network connections, and changes to file creation time," useful for security forensics.

Deployment

Distributed Cloud AIP provides an MSI file to install the Windows Agent. One-off installations are supported in the Windows Server GUI, but the Agent also supports unattended command prompt installations in "silent mode."

MANY CLOUD SECURITY THREATS ARE VISIBLE ONLY BY ESTABLISHING A PRESENCE ON THE HOST OS AND OBSERVING THE BEHAVIOR OF THE PEOPLE AND PROCESSES USING IT.

Agent Performance Considerations

Distributed Cloud AIP is constantly working to optimize the Agents to minimize overhead. But depending on what you use and how it's tuned, there can be different impacts on CPU and memory in monitored environments.

Certain Linux workloads create more kernel audit messages than others. For example: high rates of forking or execution of subprocesses. While our Agent will attempt to keep up with higher volumes of `execve` system calls, you can cap CPU utilization in these scenarios. The default setting is 40 percent utilization of the core the Agent is running on.

FIM rules are another consideration. If a filesystem is extremely busy—or if custom rules are too broadly scoped—each file or directory being watched can generate multiple events, consuming additional CPU and memory. Therefore, FIM rules should be scoped as narrowly as possible to optimize performance.

It's important to note that poor FIM performance can also be the result of file watch limits configured as part of the Linux deployment process. In these cases, it's possible to increase the limit and improve performance. On the Windows Server side, because our Agent uses a driver for FIM, we have capped memory usage at ~700 MB of kernel memory.

Network connectivity can also affect Agent performance. The Agent normally takes log entries, transforms them to JSON, and sends each object (known as "events" in our parlance) up to the Distributed Cloud AIP back end. When the Agent is unable to maintain a persistent connection, it will attempt to cache events to a local file. When connectivity resumes, the Agent continues its normal mode of operation, but with the added overhead of transmitting cached events.

Structure of an Event

The following JSON structure is an example of a Linux host server event, as it is surfaced by the Distributed Cloud AIP platform. For context, that means the event has been sent by `tsauditd` on the host OS and has then been processed on the back end where it is enriched with additional metadata such as organization_id, time_id, and other information.

The Linux host server event, as an example, would trigger an alert as the event is processed on the back end since kernel module insertion at runtime is a strong indicator of suspicious activity, but most events are far more mundane. In a well-tuned environment, typically only 1 percent of event data ever triggers an alert.

Linux Roots

```
{
        "event_type": "audit",
        "status": "success",
        "container_labels": null,
        "path": [
          "/usr/sbin/insmod",
          "/lib64/ld-linux-x86-64.so.2"
        ],
        "event_time": 1545224296000,
        "gid": 0,
        "_id": "c0c68b9a-038d-11e9-8ef5-0eb
9fbf2d436",
        "command": "insmod",
        "auid": 500,
        "container_image": null,
        "pid": 16708,
        "auser": "ec2-user",
        "organization_id": "xxxxx",
        "session": 2017,
        "exit": 0,
        "uid": 0,
        "cwd": "/home/ec2-user",
        "ppid": 16682,
        "args": [
          "insmod",
          "xpacket.ko"
        ],
        "tty": "pts0",
        "container_id": null,
        "syscall": "execve",
        "_insert_time": 1545224284685,
        "type": "start",
        "group": "ec2-user",
        "user": "root",
        "agent": {
          "name": "ip-xx-xx-xx-xx",
          "policy_id": null
        },
        "agent_id": "4f00dc5a-abca-11e8-bb29-e52888cc3b77",
        "is_agent_2": false,
        "exe": "/usr/bin/kmod",
        "arguments": "insmod xpacket.ko",
        "time_id": "c0dbe7a1-038d-11e9-ae98-7fc7b9401ba0"
}
```

Note: Schema is subject to minor changes.

Figure 1: Linux Roots Event

The following JSON structure is an example of a Windows Server host event, as it is surfaced by the Distributed Cloud AIP platform. Here, the Windows Agent has pulled a WinSec event from the Windows Event Log and has sent it up to the back end for processing and rule-based alerting.

Windows Event

```
{
       "target_user": "Angela",
       "event_time": 1564159625008,
       "win_event_id": 4776,
       "src_host": "EC2AMAZ-CIF7K66",
       "workstation": "workstation",
       "auth_package": "MICROSOFT_AUTHENTICATION_PACKAGE_
V1_0",
       "status": "0xc0000064",
       "record_number": 7193237,
       "organization_id": "xxxxx",
       "session": 0,
       "_insert_time": 1564159626028,
       "_subtype": "Credential Validation",
       "summary": "The computer attempted to validate the
credentials for an account.",
       "event_type": "winsec",
       "agent": {
         "name": "EC2AMAZ-CIF7K66",
         "policy_id": null
       },
       "placement": "Event List"
}
```

Figure 2: Windows Event

The Windows example above is a more mundane credential validation event. Out-of-the box, it should create a Low Severity alert. As rules are tuned, however, customers may choose to omit these types of alerts by excluding the rule that generates them via Amazon EC2 tags.

Further Reading

Tuning rules is beyond the scope of this document, but at a high level, all rules and processing live on the Distributed Cloud AIP back end and are completely customizable. The alerting rules are designed to prove that you have the proper monitoring and audit tracking in place to help address PCI DSS, HIPAA, ISO 27001, and other regulations.

Distributed Cloud AIP also collects other data, but it too is beyond the scope of this document. Broadly, vulnerability assessment data for Linux systems is pulled from the host's package manager and then run against the National Vulnerability Database. On both Linux and Windows Server, Distributed Cloud AIP pulls in AWS-specific data, such as EC2 resources, and integrates with CloudTrail, enabling a set of rules designed specifically to alert on suspicious API interactions between AWS services.

Supported Operating Systems

Distributed Cloud AIP currently supports various versions of the following operating systems:

- Amazon Linux
- CentOS
- Debian
- Red Hat
- Ubuntu, including support for Ubuntu 22.04
- Windows Server

Distributed Cloud AIP will never force an auto-update to agents deployed on customer systems.

Use Cases for Alerts Based on Agent Events

We have documented a set of common security use cases based on Linux Agent event data. While the docs site provides more detail and context on the alerts, you'll find a summary of some common alerting scenarios below.

1. User Access Monitoring

- Any user or group modifications
- Privilege escalations, or attempts by unapproved `sudo` users
- Unauthorized changes to monitored systems. For example, any manual change occurring outside prescribed configuration management tools.
- Audit trails of all users' TTY sessions
- Use of any privileged user accounts, such as `root`
- Abnormal login or user access attempts. For example, high rates that could indicate brute forcing of passwords

2. System Integrity Monitoring

- Unauthorized loading of kernel modules, indicating a rootkit or malware
- Deviations in use of authorized ports and services
- Any new process connection state, which could indicate command and control activity
- Abnormal process start events
- Audit trails of critical filesystem changes, including reads, transfers, and permissions

3. File Integrity Monitoring

- Modification to, or access of, critical credential files
- Changes to special system directories (`/boot/` , `/lib` , `/usr/lib` , `/bin` , `/sbin` , `/etc`) for new executables or binary replacement
- Unauthorized changes to important configuration files
- Data exfiltration activity on sensitive files, even simple "open" filesystem events

4. Network Activity Monitoring

- Any critical system services changes (NTP, DNS, Syslog) or daemon reconfiguration
- Modification of application services like web servers, database server binds, or proxy nodes
- Insecure protocol usage for system access, such as Telnet or FT

Threat Stack: Now Part of F5

Threat Stack is now F5 Distributed Cloud App Infrastructure Protection (AIP). If you'd like to learn more about this solution, the company's Security Operations Center (including Distributed Cloud AIP Managed Security Services and Distributed Cloud AIP Insights), and more, feel free to contact our cloud security and compliance experts.

Let our experts take your cloud security worries off your shoulders, so you can get down to business. To learn more or to schedule a demo, visit our website today.

[1] Red Hat Docs: security guide, "System Auditing," https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/security_guide/chap-system_auditing

DC 12.2022 | JOB-CODE-WP-950855761-Agent