How Threat Hunting Revealed Covert Sports Piracy in Data Science Environments

Assaf Morag, Threat Intelligence Director, Aqua Security's Aqua Nautilus Team

To keep up with the ever-evolving world of cybersecurity, Aqua Nautilus researchers deploy honeypots that mimic real-world development environments. During a recent threat-hunting operation, they uncovered a surprising new attack vector: threat actors using misconfigured servers to hijack environments for streaming sports events. By exploiting misconfigured JupyterLab and Jupyter Notebook applications, attackers drop live streaming capture tools and duplicate the broadcast on their illegal server, thus conducting stream ripping. In this blog, we explain how our threat hunting operation helped us uncover this and how we analyzed this attack using Aqua Tracee and Traceeshark.

Nautilis' Threat Hunting Efforts

When utilizing honeypots to collect threat intelligence, you assume that any event is malicious. In reality, there are many scanners that trigger the honeypots, script kiddies that trigger events with their curiosity, or trivial tools and failed attack attempts that exploit initial access but fail to mature to a full-blown attack. Strong automation and machine learning were tailored to distinguish between interesting and non-interesting events. But sometimes we miss, and when that happens, we utilize threat hunting as a compensative measurement.

In our recent threat hunting operation, we focused on analyzing inbound network traffic and dropped and executed binaries within containerized environments to uncover potential hidden security breaches. Our honeypots generate thousands of events per day, automatically saved to various data environments for storage (data lake) and analysis (document database and data warehouse). By leveraging our data warehouse, we cross-referenced some of our signatures to link suspicious binaries with network events, revealing patterns indicative of illicit activity. Once these connections were established, we honed in on specific events and sessions tied to these anomalies, enabling us to isolate, examine, and address potential security threats in real time.

We found several dozen events which indicated that a benign tool was dropped and executed. The tool 'ffmpeg' is an open source software suite used for recording, converting, and streaming audio and video. It supports a wide range of multimedia formats and is widely utilized for video processing, compression, and live streaming applications. Threat intelligence platforms such as Virus Total indicated that this is not a malicious tool, and indeed it is not considered malicious or potentially unwanted, thus this was never classified as an attack, at least until now.

About Jupyter Lab & Jupyter Notebooks

JupyterLab and Jupyter Notebook are two powerful interactive environments for data science. Many organizations utilize these tools for their everyday data operations, but there are some potential risks, if not properly secured. Often managed by data practitioners who may lack awareness of common misconfigurations. Including connecting the server to the internet with open access without authentication, which allows unauthorized users to run code. Additionally, exposing the Jupyter stack to the internet without firewalls, making it vulnerable to attacks. Token mishandling is another issue, as exposed tokens can grant full access. Based on Shodan there are ~15,000 Jupyter servers connected to the internet, while ~150 (1%) enable remote code execution. Nautilus' analysis shows some private personal Jupyter notebooks, as well as corporate and startup whose servers are exposed to anyone, actively exploited.

Running the Jupyter stack with restricted IPs, strong authentication, HTTPS, and token management can mitigate these risks, helping secure sensitive data and code.

About Illegal Live Streaming of Sports Events

Illegal live streaming of sports events is a growing threat to the industry, cutting into revenue streams for leagues, broadcasters, and legitimate platforms. With high-speed internet and accessible streaming tools, unauthorized broadcasts have become widespread, impacting not only big leagues but also smaller teams that rely on paid viewership.

To counter this, sports organizations use advanced technologies like AI-based detection, watermarking, and digital rights management (DRM) to track and shut down illegal streams in real time. Legal actions and collaborations with governments help enforce copyright, while public awareness campaigns aim to shift viewer behavior.

The Attack Flow

Our Jupyter Lab and Jupyter Notebook honeypots reveal both vulnerabilities and weak passwords. In this case, threat actors exploited unauthenticated access to Jupyter Lab and Jupyter Notebook to establish initial access and achieve remote code execution.

First, the attacker updated the server, then downloaded the tool *ffmpeg*. This action alone is not a strong enough indicator for security tools to flag malicious activity. Next, the attacker executed *ffmpeg* to capture live streams of sports events and redirected them to their server.

Below you can see the entire attack flow:

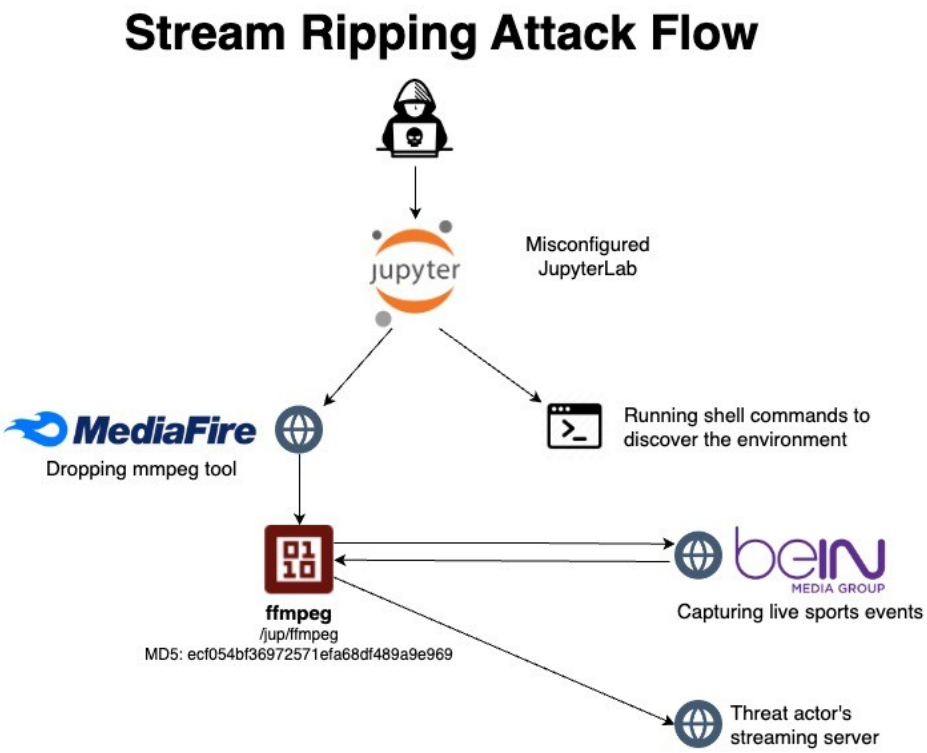


Figure 1: The entire attack flow

This straightforward attack is easy to overlook. While the immediate impact on organizations might appear minimal (though it significantly affects the entertainment industry), it could be dismissed as merely a nuisance.

However, it's crucial to remember that the attackers gained access to a server intended for data analysis, which could have serious consequences for any organization's operations. Potential risks include denial of service, data manipulation, data theft, corruption of AI and ML processes, lateral movement to more critical environments, and, in the worst-case scenario, substantial financial and reputational damage.

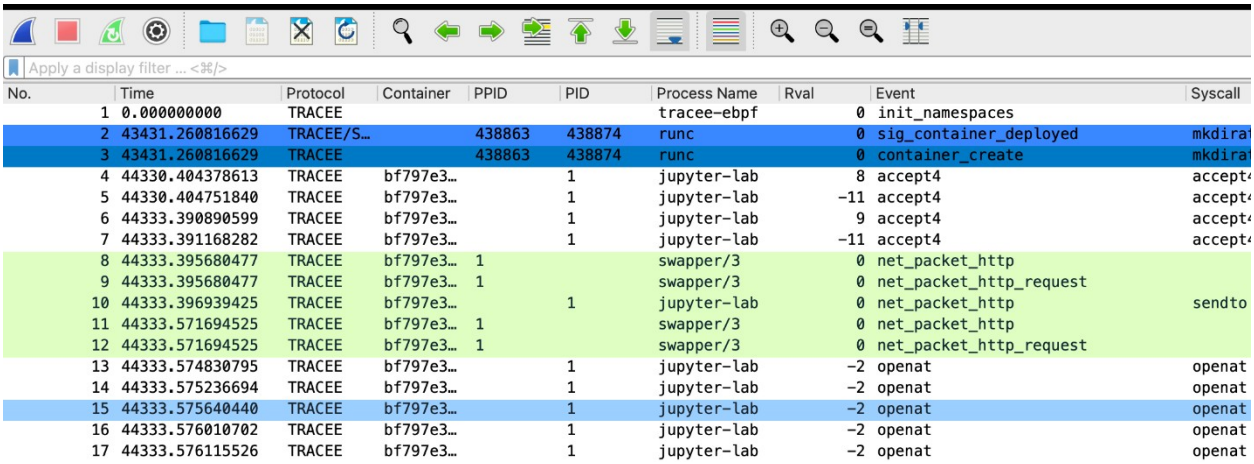
Analyzing the Attack with Aqua Tracee and TraceeShark

Aqua Tracee is a runtime security and forensics tool for Linux, utilizing eBPF technology to trace systems and applications at runtime, analyze kernel level events to detect suspicious behavioral patterns, and capture forensics artifacts. You can read about its evolution in our blog "The Story of Tracee: The Path to Runtime Security Tool".

Traceeshark brings Linux runtime security monitoring to Wireshark, enabling cross-platform analysis of Tracee events. With Wireshark's interactive filtering and data aggregation, large Tracee datasets become manageable. Users can capture Tracee events in Wireshark locally, semi-locally via Docker, or remotely over SSH. This integration allows combined system and network analysis, adding context to host-based network monitoring. You can read more about Traceeshark in our blog "Go deeper: Linux runtime visibility meets Wireshark".

Tracee enables capturing Linux events on the server, including network activity, files, and dumps of suspicious memory regions. We consolidate the events and network data generated by Tracee into a single Wireshark-compatible .pcapng file to begin our investigation

We then upload the `.pcapng` file to Traceeshark, a modified version of Wireshark tailored for our analysis. As shown in Figure 2 below, the Wireshark framework has been enhanced to support Tracee's data points, including 'protocol type,' 'container ID,' process and parent process IDs, among others.


Apply a display filter ... <⌘/>

No.	Time	Protocol	Container	PPID	PID	Process Name	Rval	Event	Syscall
1	0.000000000	TRACEE				tracee-ebpf	0	init_namespaces	
2	43431.260816629	TRACEE/S…		438863	438874	runc	0	sig_container_deployed	mkdira
3	43431.260816629	TRACEE		438863	438874	runc	0	container_create	mkdira
4	44330.404378613	TRACEE	bf797e3…		1	jupyter-lab	8	accept4	accept
5	44330.404751840	TRACEE	bf797e3…		1	jupyter-lab	-11	accept4	accept
6	44333.390890599	TRACEE	bf797e3…		1	jupyter-lab	9	accept4	accept
7	44333.391168282	TRACEE	bf797e3…		1	jupyter-lab	-11	accept4	accept
8	44333.395680477	TRACEE	bf797e3…	1		swapper/3	0	net_packet_http	
9	44333.395680477	TRACEE	bf797e3…	1		swapper/3	0	net_packet_http_request	
10	44333.396939425	TRACEE	bf797e3…		1	jupyter-lab	0	net_packet_http	sendto
11	44333.571694525	TRACEE	bf797e3…	1		swapper/3	0	net_packet_http	
12	44333.571694525	TRACEE	bf797e3…	1		swapper/3	0	net_packet_http_request	
13	44333.574830795	TRACEE	bf797e3…		1	jupyter-lab	-2	openat	openat
14	44333.575236694	TRACEE	bf797e3…		1	jupyter-lab	-2	openat	openat
15	44333.575640440	TRACEE	bf797e3…		1	jupyter-lab	-2	openat	openat
16	44333.576010702	TRACEE	bf797e3…		1	jupyter-lab	-2	openat	openat
17	44333.576115526	TRACEE	bf797e3…		1	jupyter-lab	-2	openat	openat

Figure 2: Traceeshark main view

As can be seen in Figure 3, there are over 8,000 events, thus going over them one-by-one can be tricky and time consuming. We will leverage various analytics capabilities to gain timely insights into this attack. Using the 'statistics' options, we can assess the volume of signatures and events, indicating that this is a relatively small session with limited potential for a significant attack. However, we will proceed with our analysis to ensure thorough evaluation.

Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
Signatures	49				0.0000	100%	0.0900	44369.066
Severity 3	0				0.0000	0.00%	-	-
Severity 2	5				0.0000	10.20%	0.0100	44494.690
Severity 1	9				0.0000	18.37%	0.0300	44369.066
Severity 0	35				0.0000	71.43%	0.0600	44369.066
Other Severities	0				0.0000	0.00%	-	-
Events	8775				0.0002	100%	6.5200	44357.860
task_rename	15				0.0000	0.17%	0.0400	44369.066
shared_object_loaded	71				0.0000	0.81%	0.3100	44529.593
set_fs_pwd	8				0.0000	0.09%	0.0400	44529.670
security_task_setrlimit	21				0.0000	0.24%	0.2100	44529.595
security_socket_create	44				0.0000	0.50%	0.1500	44529.599
security_socket_connect	37				0.0000	0.42%	0.1000	44529.599
security_inode_unlink	4				0.0000	0.05%	0.0200	44535.495
security_inode_rename	8				0.0000	0.09%	0.0400	44360.159
security_file_open	7551				0.0002	86.05%	6.5200	44357.860
sched_process_fork	38				0.0000	0.43%	0.1000	44658.249
sched_process_exit	15				0.0000	0.17%	0.0300	44369.152
sched_process_exec	15				0.0000	0.17%	0.0400	44369.066
rt_sigaction	27				0.0000	0.31%	0.0700	44529.592
prctl	3				0.0000	0.03%	0.0300	44529.599
openat	295				0.0000	3.36%	0.3300	44529.593
open	2				0.0000	0.02%	0.0100	44612.074
net_packet_http_request	152				0.0000	1.73%	0.0600	44356.310
net_packet_http	304				0.0000	3.46%	0.1200	44356.310
net_packet_dns_response	2				0.0000	0.02%	0.0200	44415.759
net_packet_dns_request	2				0.0000	0.02%	0.0200	44415.750
net_packet_dns	4				0.0000	0.05%	0.0400	44415.750
magic_write	8				0.0000	0.09%	0.0100	44360.460
init_namespaces	1				0.0000	0.01%	0.0100	0.000
dup2	8				0.0000	0.09%	0.0400	44369.069
do_truncate	3				0.0000	0.03%	0.0100	44361.256
container_create	1				0.0000	0.01%	0.0100	43431.261
commit_creds	4				0.0000	0.05%	0.0400	44529.600
arch_prctl	15				0.0000	0.17%	0.0400	44369.067
accept4	117				0.0000	1.33%	0.1200	44356.304

Figure 3: Number of packets

So far, the evidence may not seem compelling. However, as shown in Figure 4, the situation becomes suspicious when observing the precess tree suggesting of numerous `ffmpeg` execution commands, especially with the unusual pattern of IP addresses involved. If you're familiar with `ffmpeg` and its typical operations, you'll immediately recognize the nature of this abnormal server activity. If not, these several command executions related to traffic would still raise significant red flags.

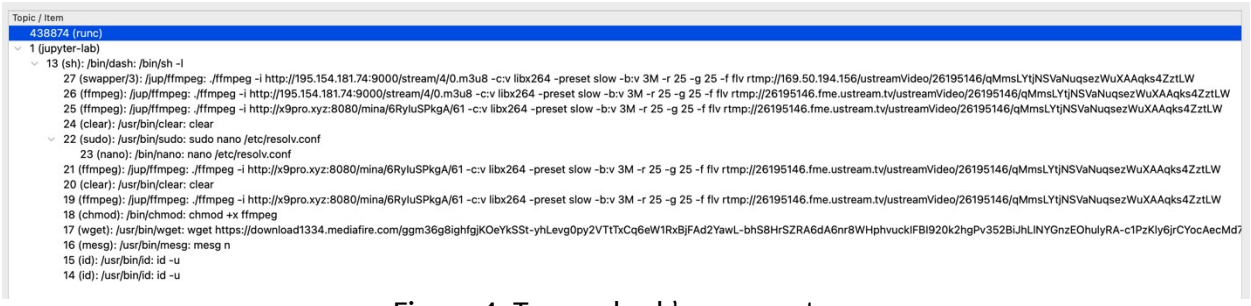


Figure 4: Traceeshark's process tree

With the dedicated filter we integrated into Traceeshark, we can now delve deeper into analyzing this attack. Using the 'Container' filter, we can isolate events at the container level, viewing only 'Events,' 'Network' packets, or 'Signatures' that were triggered. In this instance, we'll apply the 'Important' filter, configured to highlight "Significant" events and signatures identified based on our expertise.

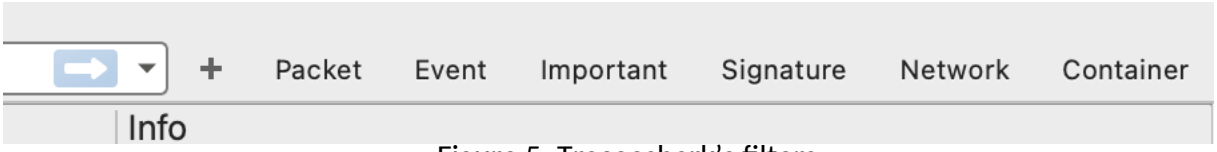


Figure 5: Traceeshark's filters

In this case, this is a JupyterLab server with misconfiguration. It is connected to the internet with no authentication required, so if an attack puts in his browser "http[:]//IP-Address[:]8888/ /tree?", allowing a remote code execution.

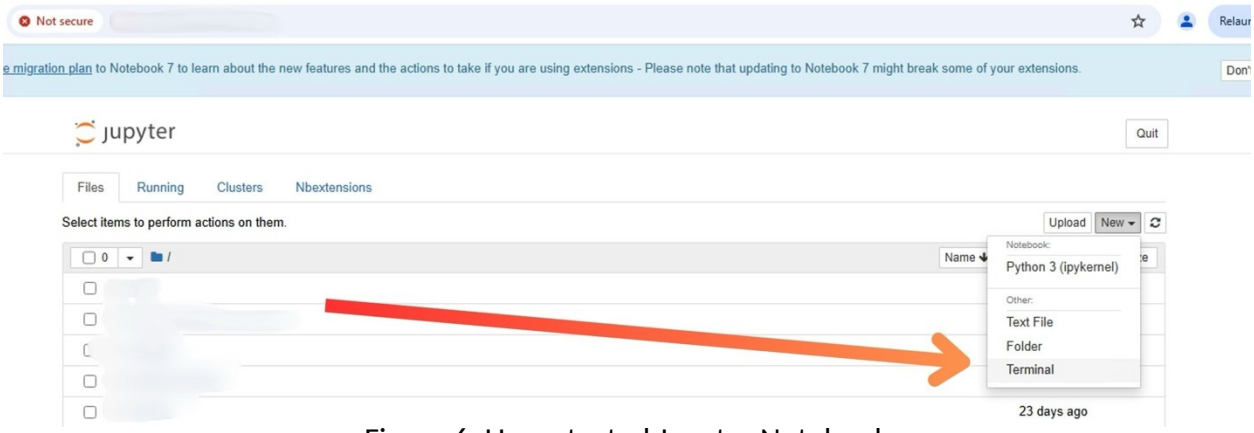


Figure 6: Unprotected Jupytar Notebook

Continuing with Traceeshark we can observe the discovery commands of the threat actor as illustrated in Figure 7 below.

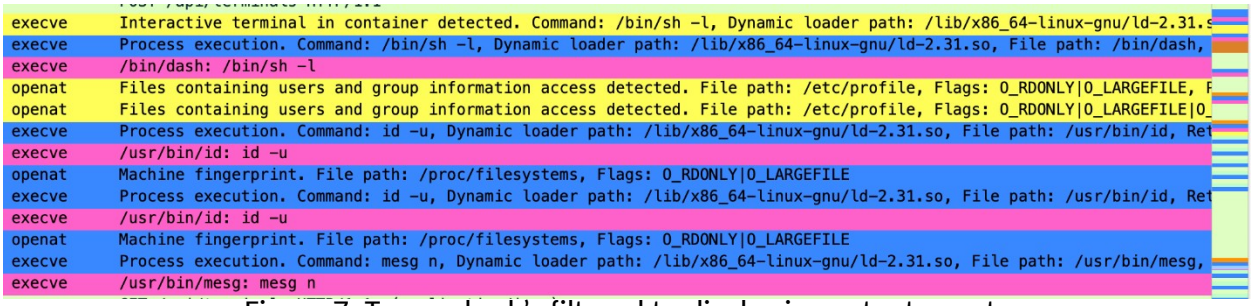


Figure 7: Traceeshark's filtered to display important events

Next, we observe the download of `ffmpeg` (MD5: ecf054bf36972571efa68df489a9e969) from the MediaFire file-sharing website. This download source adds another layer of suspicion, as it suggests the file may not be an official or trusted version, potentially carrying malicious modifications or being used in an unauthorized manner. But inspection of the file in Virus Total and IDA suggests, it's benign.

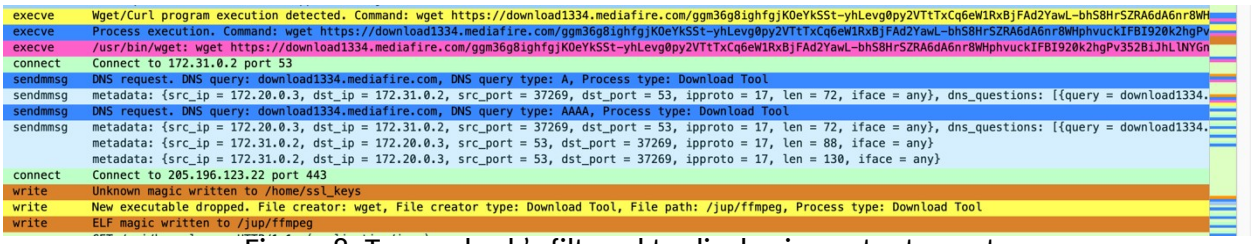


Figure 8: Traceeshark's filtered to display important events

Wireshark allows users to double-click a packet to examine its contents, and Traceeshark extends this functionality by displaying the content of Tracee logs. In Figure 10 below, we see the executed command in detail. Here, the tool `./ffmpeg` is launched, with the streaming source (`-i`) set to `x9pro.xyz`, accompanied by flags to control the streaming speed and minimize detection. The output is directed to `ustream.tv`, revealing the intent to stream captured content to an external platform discreetly.

Ustream offers various monetization options for content creators. Earnings primarily come from ad revenue, where ads placed on videos generate income per view or click, as well as from paid subscriptions for exclusive content. During live streams, creators can also earn through donations or tips from viewers. To qualify for these earnings, creators often need to meet minimum requirements for followers or view counts. Unfortunately, threat actors exploit similar methods by stream-ripping sports event feeds and illegally broadcasting them on their own channels to profit from unauthorized views and ad revenue.

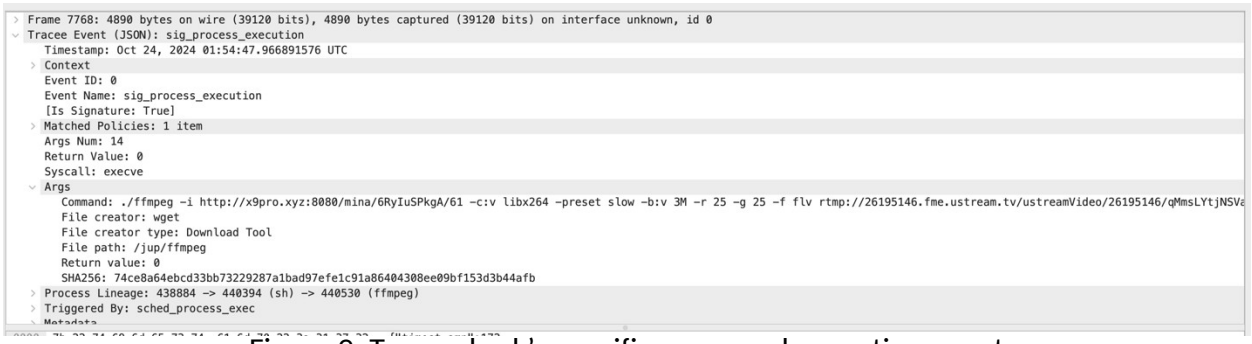


Figure 9: Traceeshark's specific command execution event

After analyzing the sources of live streaming the threat actors tried capturing via our server, we concluded that threat actors targeted live streaming broadcastss of the Qatari beIN Sports network. The IP address they used was from Algerian AS (41.200.191.23), indicating that they might be of Arab speaking origin as well.

Figure 10: The targeted source broadcasted the UEFA Champions League fixture between Shakhtar Donetsk and BSC Young Boys played on November 6, 2024

Mapping the Campaign to the MITRE ATT&CK Framework

Our investigation showed that the attackers have been using some common techniques throughout the campaign. Here we map each component of the attack to the corresponding techniques of the MITRE ATT&CK framework:

Initial Access	Execution	Exfiltration	
Exploit Public-Facing Application (T1190)	Command and scripting interpreter: Unix Shell (T1059.001)	Exfiltration Over Alternative Protocol: Exfiltration Over Unencrypted/Encrypted Non-C2 Protocol (T1048.003)	Res

Initial Access

- Exploit Public-Facing Application: Attackers exploited misconfigured JupyterLab and Jupyter Notebook applications, gaining unauthenticated access to establish initial access to the development environment.

Execution

- Command and Scripting Interpreter - Unix Shell: The attackers executed commands through the Jupyter Notebook to install and run ffmpeg.

Exfiltration

- Exfiltration Over Alternative Protocol - Exfiltration Over Unencrypted/Encrypted Non-C2 Protocol: The attackers exfiltrated video content through ffmpeg streams directed to external destinations.

Impact

- Resource Hijacking: The attackers use the victims bandwidth to transfer streaming data.

Detection and Mitigation

In this blog, we explore how behavioral analysis, combined with proactive threat hunting, plays a crucial role in identifying hidden attacks and potentially unwanted activities.

Traditional security tools often miss subtle indicators, especially in complex environments like JupyterLab and Jupyter Notebook, where legitimate tools can be used for unauthorized purposes. By using behavioral analysis to spot anomalies—such as the unusual deployment and execution of `ffmpeg` for live-stream capture—our team uncovered covert sports piracy operations that bypassed standard alerts.

This approach, leveraging tools like Aqua Tracee and Traceeshark, highlights how advanced monitoring of patterns and behavioral indicators can detect misuse that may appear benign at first glance but could lead to significant security and operational impacts.

Indications of Compromise (IoCs)

Type	Value	Comment
IP Addresses		
IP Address	167.99.93.212	Attacker IP
IP Address	41.200.191.23	Attacker IP