# IOC (Indicators of compromise)

Indicators of compromise (IOCs) are “pieces of forensic data, such as data found in system log entries or files, that identify potentially malicious activity on a system or network.” Indicators of compromise aid information security and IT professionals in detecting data breaches, malware infections, or other threat activity. By monitoring for indicators of compromise, organizations can detect attacks and act quickly to prevent breaches from occurring or limit damages by stopping attacks in earlier stages.

[What are Indicators of Compromise? | Digital Guardian](https://digitalguardian.com/blog/what-are-indicators-compromise)

## INDICATORS OF COMPROMISE VS. INDICATORS OF ATTACK

Indicators of attack are similar to IOCs, but rather than focusing on forensic analysis of a compromise that has already taken place, indicators of attack focus on identifying attacker activity while an attack is in process. Indicators of compromise help answer the question “What happened?” while indicators of attack can help answer questions like “What is happening and why?” A proactive approach to detection uses both IOAs and IOCs to discover security incidents or threats in as close to real time as possible.

## What is an artifact in cyber security?

Artifacts are tracks that get left behind. You could associate them with the footprints of the end-user or hacker. However, end-users are often unaware that artifacts exist. Like permanent footprints, they are challenging to manipulate. As a result, artifacts help cyber security consultants in their role of uncovering the root causes of a data breach and the threat actors involved.

## Malware

Malware (short for “malicious software”) is a file or code, typically delivered over a network, that infects, explores, steals or conducts virtually any behavior an attacker wants. And because malware comes in so many variants, there are numerous methods to infect computer systems.

### Ransomware

Ransomware is a type of malicious software (malware) that threatens to publish or blocks access to data or a computer system, usually by encrypting it, until the victim pays a ransom fee to the attacker. In many cases, the ransom demand comes with a deadline. ... Ransomware attacks are all too common these days.

## Reactive Microservices

* Reactive microservices are FAST
* each service manages its own state in memory
* no need to go to a database for state
* Each service can communicate directly with other services asynchronously
* Databases can be accessed asynchronously for event sourcing / event logging
* Services automatically scale up to handle peak workloads and scale down when needed.
* Services automatically rebalances workload and self heal in case of failure

[What is a reactive microservice? – O'Reilly](https://www.oreilly.com/content/what-is-a-reactive-microservice/)

The **microservices architecture** breaks the software application into lightweight and independent pieces of logical relevance; this formation gives indispensable benefits to the modern software industry. The Benefits involve faster and easier delivery of new products and features, easy scaling out to serve more customers, and introducing failure isolation - the system is supposed to remain operational despite failure.

**Reactive Microservices** design suggests additional isolation between the parts of the application so that every microservice is dealt with as an external SaaS or PaaS product. This signifies the most secure way against internal and external data piracy, the highest availability and resiliency, and encourages team autonomy inside the enterprise.

Reactive microservices architecture is a specific type of microservices architecture. The reactive property enables a service to be elastic (compute resources can scale up and down on the cloud), resilient (if a node fails, it can self-heal) and responsive (high availability / low latency). The key to reactive microservices is asynchronous message passing between services.

## Multithreading vs Multiprocessing

**Multithreading** is a technique where multiple threads are spawned by a process to do different tasks, at about the same time, just one after the other. This gives you the illusion that the threads are running in parallel, but they are actually run in a concurrent manner. In Python, the Global Interpreter Lock (GIL) prevents the threads from running simultaneously.

**Multiprocessing** is a technique where parallelism in its truest form is achieved. Multiple processes are run across multiple CPU cores, which do not share the resources among them. Each process can have many threads running in its own memory space. In Python, each process has its own instance of Python interpreter doing the job of executing the instructions.

[Difference Between Multithreading vs Multiprocessing in Python - GeeksforGeeks](https://www.geeksforgeeks.org/difference-between-multithreading-vs-multiprocessing-in-python/?ref=rp)

### Coroutine

coroutine is a function that can suspend its execution before reaching return, and it can indirectly pass control to another coroutine for some time.

[Asyncio Tutorial: Async Programming in Python](https://djangostars.com/blog/asynchronous-programming-in-python-asyncio/)

### Python Generators

[What Are Python Generators?](https://realpython.com/lessons/what-are-python-generators/)

## Headers

The REST headers and parameters contain a wealth of information that can help you track down issues when you encounter them. HTTP Headers are an important part of the API request and response as they represent the meta-data associated with the API request and response. Headers carry information for:

* Request and Response Body
* Request Authorization
* Response Caching
* Response Cookies

call apoc.trigger.add('kafka', "UNWIND apoc.trigger.nodesByLabel($createdNodes,'Student') AS node Call streams.publish('registered\_user', 'working!!!')", {phase:'after'})

call apoc.trigger.add("kafkaa",

"UNWIND apoc.trigger.nodesByLabel($createdNodes, 'Student') AS node

CALL streams.publish('registered\_user', 'working!!!') return null", {phase:'after'})

call apoc.trigger.add("kafkaa",

"UNWIND $createdNodes as n

MATCH (n)

with distinct labels(n) as label

where 'Student' in label

call streams.publish('registered\_user', 'working!!!') return NULL", {phase:'after'})

For opendistro-kibana

kubectl port-forward services/elasticsearch-opendistro-es-kibana-svc 9700:443