Domain Knowledge

🔐 IOC = *Indicator of Compromise*

A clue that something bad (like a cyber-attack or malware) has occurred or is about to happen.

Examples of IOCs:

- Malicious IP addresses
- Malicious URLs
- File hashes (MD5, SHA256)
- Suspicious domains

👉 In your project, **every item stored and shown is an IOC**.

🌐 IP Address (Individual IPs)

An address that identifies a device on the internet.

Example:

- 192.168.0.1 (home router)
- 8.8.8.8 (Google DNS)

In the feeds:

- Blocklist and some Spamhaus entries include individual IPs that are known attackers or scanners.

🌍 IP Subnet (CIDR Block)

A range of IP addresses grouped together.

Example:
192.168.1.0/24 means:

- From 192.168.1.0 to 192.168.1.255
- Used to block or monitor a **whole group of IPs**

Spamhaus feed contains many subnets like:

43.242.128.0/22 ; SBL123456 — Known botnet range

You extract just the subnet from the line.

🦠 Malicious URL

A link that hosts or distributes malware, phishing, or exploit kits.

Examples:

- http://badguy.ru/stealer.exe
- https://malicious.site/payload.zip

In this project, you're filtering URLs that end in .exe, .zip, etc.

🛡️ Spamhaus

A trusted cybersecurity organization that tracks:

- Botnets
- Spam networks
- Malware domains

URL: http://www.spamhaus.org/drop/drop.txt

- Their **DROP list** contains IPs/subnets you should never route traffic to.
- Spamhaus is widely respected in the industry.

🔒 Blocklist.de

A German open-source community service

- Collects logs from people's servers
- Publishes IPs involved in brute-force attacks

URL: http://www.blocklist.de/lists/apache.txt

- Contains **individual IPs** that attacked Apache web servers.

🗄️ Normalization

Converting data from different formats into one **standard schema**.

```
{
  "type": "ip" or "url",
  "value": "8.8.8.8" or "http://bad.site/drop.exe",
  "source": "blocklist" or "spamhaus" or "digitalside",
  "timestamp": "2025-08-01T07:52:10.000Z"
}
```

🔎 What is an Octet?

In the context of IP addresses:

An **octet** is one of the **four numbers** in an IPv4 address.

Each octet is:

- A number between **0 and 255**
- Represents **8 bits** (hence "oct" = 8)

🧩 What is CIDR?

- CIDR = Classless Inter-Domain Routing
- A way to represent IP address ranges.
- 192.168.0.0: base IP
- /24: number of bits used for the network part
  (this means IPs from 192.168.0.0 to 192.168.0.255)

Approach to calculate confidence

🧠 Step-by-Step Breakdown: calculate_confidence()

✅ 1. Start with base confidence by source

python

CopyEdit

```
source_confidence = {
    'blocklist': 0.8,
    'spamhaus': 0.9,
    'digitalside': 0.7
}
```

🔍 Explanation:

Each threat intelligence source is assigned a base score reflecting how much you trust the data from that source.

Source	Why Chosen	Base Confidence
spamhaus	Reputable and curated list	0.9 (Very High)
blocklist	Known attackers or blacklists	0.8
digitalside	Public data, somewhat noisy	0.7
others	Unknown/untrusted sources	0.5 (Default)

Default (0.5) is used for unrecognized or low-trust sources. It's a neutral midpoint, assuming equal chance of false positives and true positives.

✅ 2. Adjust based on IOC type

After setting the base score, you boost it slightly depending on how dangerous or trustworthy the format/content of the IOC is.

✅ IP / CIDR Block Adjustment

python

CopyEdit

```
if '/' in ioc['value']:
    base_score += 0.05
```

- If the value contains '/', it's a CIDR block (e.g., 192.168.0.0/24), not a single IP.
- CIDRs often represent entire malicious networks, not just individual compromised hosts.
- So you trust them more = slightly increase confidence (+0.05)

✅ URL Adjustments Based on File Type

python

CopyEdit

```
suspicious_extensions = ['.exe', '.zip', '.rar', '.bat', '.scr']
if any(ioc['value'].endswith(ext) for ext in suspicious_extensions):
    base_score += 0.1
```

- These extensions are commonly associated with malicious downloads:
  - .exe – Windows executable
  - .zip / .rar – Compressed packages often hiding malware
  - .bat / .scr – Scripts or screensavers used in attacks
- If a URL ends with one of these, it's very likely to be malicious ⇒ confidence gets a bigger boost (+0.1)

✅ 3. Cap the Score

python

CopyEdit

return min(base_score, 1.0)

- Ensures confidence score never exceeds 1.0
- Prevents overconfidence when multiple conditions stack up
- Keeps it mathematically bounded for downstream usage (e.g., filtering, thresholds)

🤔 Why these default values?

- They are heuristics — informed estimates based on:
  - Industry trust level in various feeds
  - Malware trends
  - Empirical threat intelligence practices

Default Scores Used:

Condition	Value	Why
Unknown source	0.5	Baseline — "I don't know" trust level
CIDR boost	+0.05	CIDRs are broader = more dangerous
Suspicious file in URL	+0.1	Known malware delivery vectors
Final max	1.0	To prevent overscoring

✅ Summary

The function builds a trust score by:

- Starting with where the data came from
- Tweaking it based on what the data looks like
- Ensuring the score stays within a safe range

This makes the IOC evaluation scalable, consistent, and reasonably accurate, even when feeds come from many varied sources.

Let me know if you want to tweak the logic for emails, domains, or enrichments like geolocation or ASNs!

No matter how many adjustments are made, the score will never exceed 1.0.

✅ Which Confidence Score Value is "Good"?

It depends on **how you're using the score** — but here's a general interpretation scale:

Confidence Score	**Meaning**	**Should You Trust the IOC?**
> 0.85	**High confidence** (Very likely malicious)	✅ Yes — strong candidate for blocking
0.7 – 0.85	**Medium confidence**	⚠️ Possibly malicious — investigate
0.5 – 0.7	**Low confidence**	❓ Suspicious, but uncertain
< 0.5	**Untrusted or unknown**	🚫 Probably not reliable enough

---

🔍 Suggested Actions Based on Score

Score Range	**Suggested Action**
>= 0.85	Add to blocklists or alert analysts
0.7 - 0.84	Log and enrich, maybe alert on repeated sightings
0.5 - 0.69	Flag for further correlation (don't alert yet)
< 0.5	Ignore or suppress unless corroborated by other data

💡 When tuning your pipeline:

- **Use 0.8 as your block threshold** if you're dealing with real-time blocking.
- **Use 0.6 for alerting** if you prefer to investigate before acting.
- Tune these values based on:
  - Your **false positive tolerance**
  - The **type of environment** (corporate network vs threat research)