UNIT 42
BY PALO ALTO NETWORKS

Indicator Wranglin’ - An Approach to Dynamically Typing IOCs with Poor Data Context

Noah Dunn

CYBER THREAT ALLIANCE
Threat Intelligence Practitioners’ Summit

paloalto NETWORKS

whoami

- Noah Dunn, Senior Security Automation Engineer for Unit 42 at Palo Alto Networks
- Former US Fed Contractor
- OSCP, GREM, ID10T, etc...
- From Ohio, United States
- Fake food critic on Google Reviews
- Avid Board Game player

Objectives for This Talk

- Discuss a data stack agnostic pipeline to process IOCs
- Provide information in a manner that can be implemented in other tech stacks and organizations
- Explain an example of how Unit 42 uses this technology to the benefit of our Threat Intelligence

Our Organization's Problem

- My engineering organization services two separate stakeholders, each with their own goals and approaches to work: **Threat Intelligence** and **Incident Response**
- We use Incident Response data that is gathered in order to inform the research of Threat Intelligence

- **The Problem**: Incident Response processes and stores data in a way that is not conducive for Threat Intelligence research

Our Organization's Problem (cont.)

Threat Intelligence Objectives

- Provide concise descriptions of threat actors and malware variants
- Work through attribution problems, very concerned about who is responsible and why
- Requires **accurate** and **concise** data for research publications
- Less concerned with data quantity, but more concerned with **data quality**

Incident Response Objectives

- Help clients out with triaging and analyzed compromised systems
- Gather as many artifacts and IOCs as possible to build and understand how systems were compromised, and why (financial, political, etc.)
- Requires **complete** and **verbose** data for collection and analysis
- Less concerned with data quality, but more concerned with **data quantity**

Our Organization's Problem (cont.)

- Due to Incident Response prioritizing a complete collection of information, they provide a full set of data that is not necessarily concise
- Incident Response data is critical to Threat Intelligence, but Threat Intelligence does not have the time to filter out, deduplicate, and determine what is relevant to their cause
- In addition, we have no context to indicate what indicators actually are (IP? URL? Hash?)
- That's where engineering automation comes in with the **Two-Phase Identification Solution**

Fanging And Defanging Overview

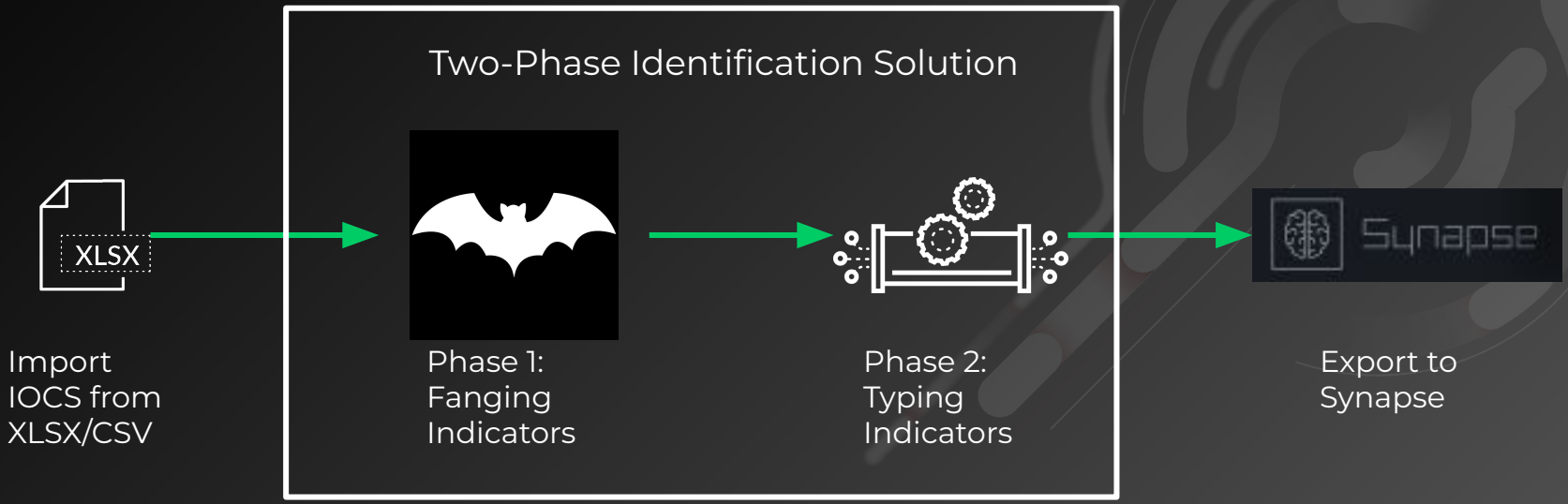
As an Overview, to clarify some terms for the sake of this discussion:

- **Fanging IOCs** - Presenting IOCs in a format that can easily be clicked on and lead to malicious places
- **Defanging IOCs** - Presenting Domains/IPs/Emails/URLs in a way so they can't be easily clicked on and accessed
  - Everyone defangs in a different style, but from our data set there are a few patterns that capture *almost all* use cases
  - Refanging before sending to a machine operated data engine is critical to ensure known patterns are matched (there's too many patterns to match against if we don't)

Fanging and Defanging Overview (cont.)

- For this pipeline, we follow the rule of **Ingesting Defanged** and **Exporting Fanged**
- Data sources are normally heavily accessed and edited by real, human, people
- Data engines that data is exported to is normally pure machine automation
- We can ensure data quality in this way that does not put IR Consultants or Threat Intelligence researchers at risk of accidental clicks

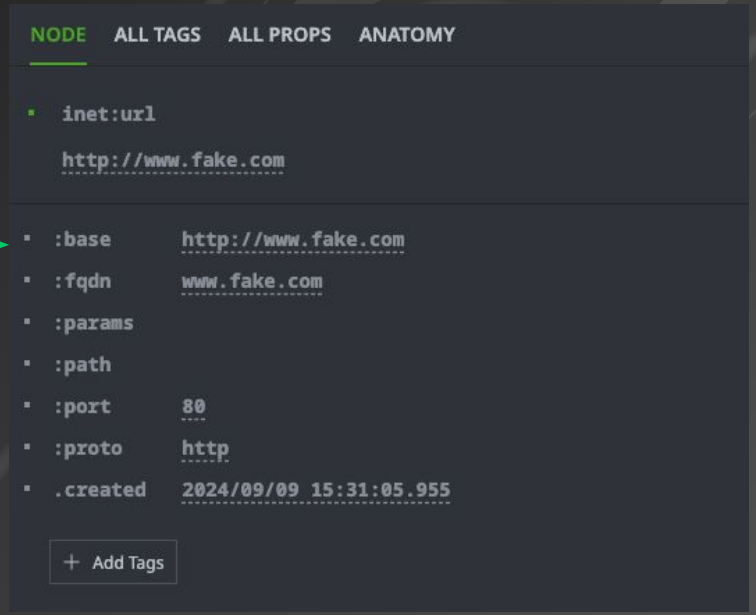
The Solution



Note: The Two-Phase solution is agnostic. The import and export can be swapped for any appropriate combination with the correct support.

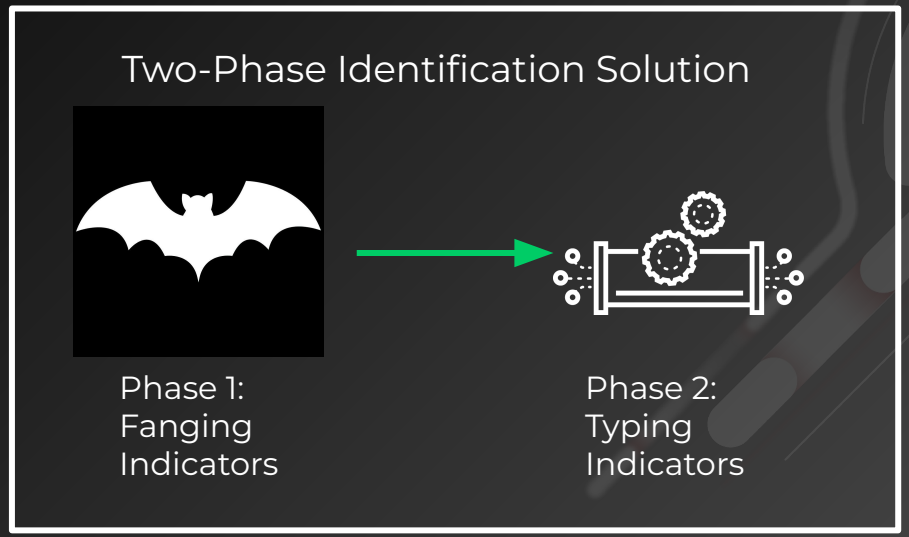
The Goal

Indicator
www.fake[.]com
alsofake/.exe
definitely_fake/still_fake/more_fake*sh
127.0.0.1
1.1.1.1
http[:]//wowtheresfakeeverywhere.net



 |

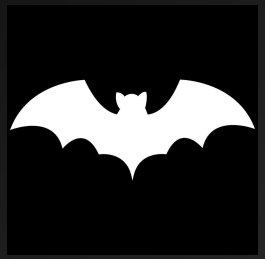
Two Phase Solution Overview



Two Phase Solution Overview

Input

www[.]fake.com



Phase 1:
Fanging
Indicators

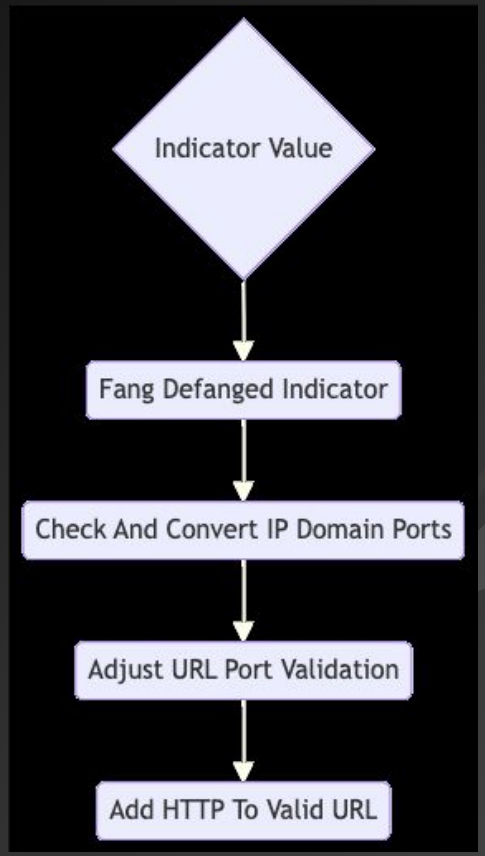
http://www.fake.com

Phase 2:
Typing
Indicators

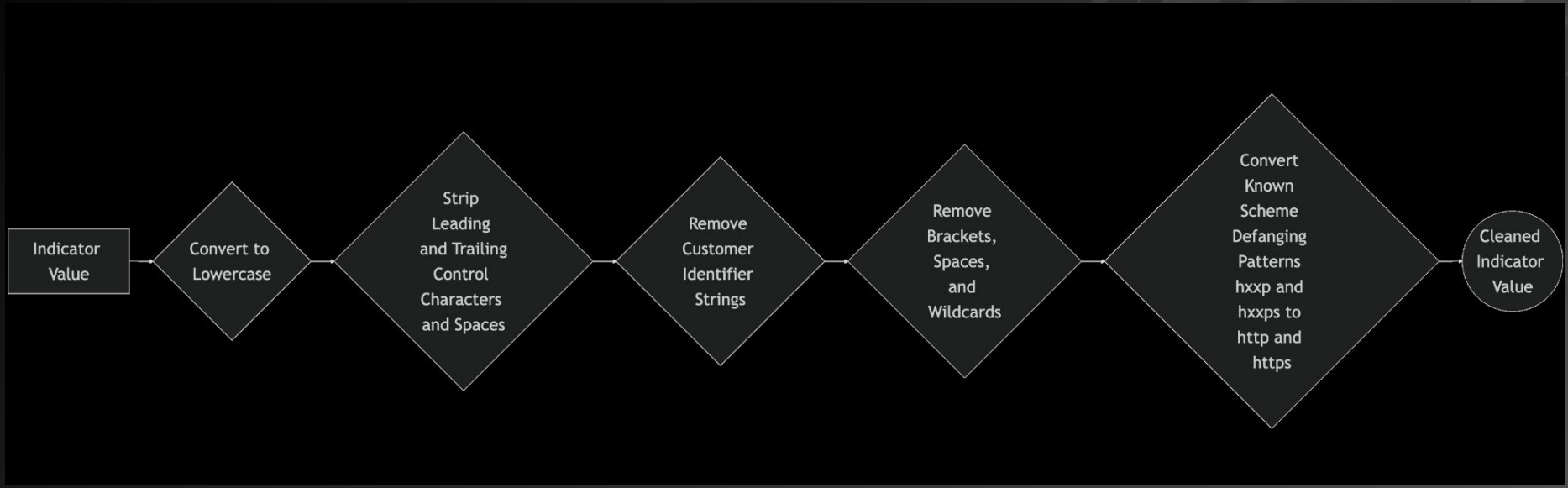
Output

Indicator: http://www.fake.com
Type: URL

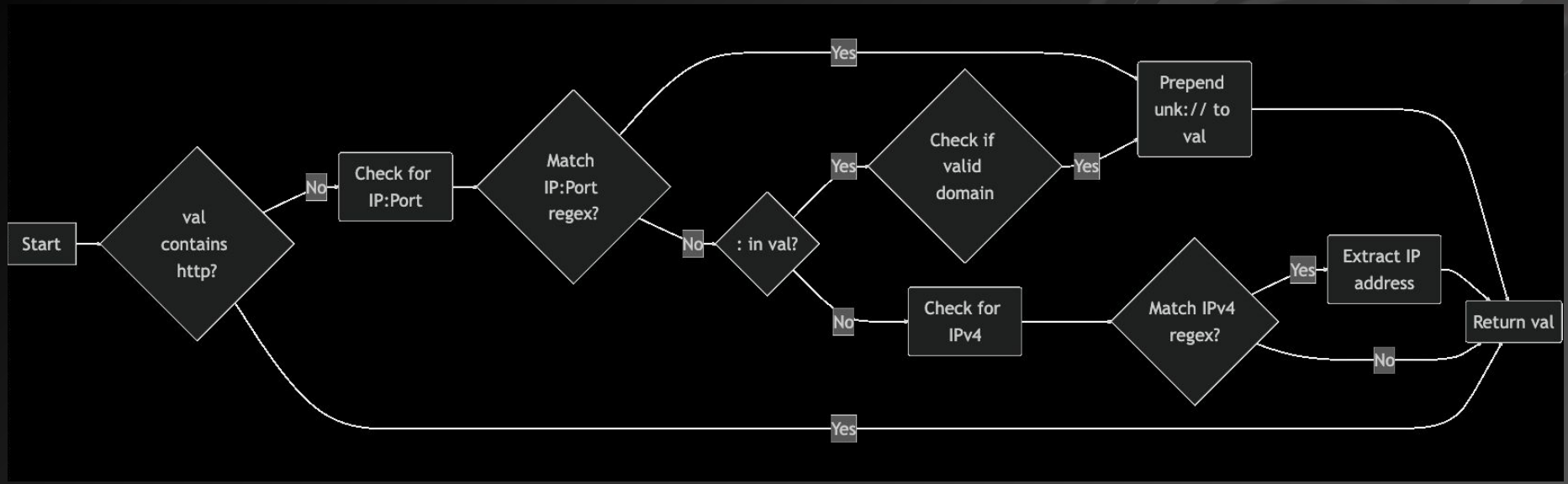
Phase 1: Fanging Indicators - Process Indicator Value



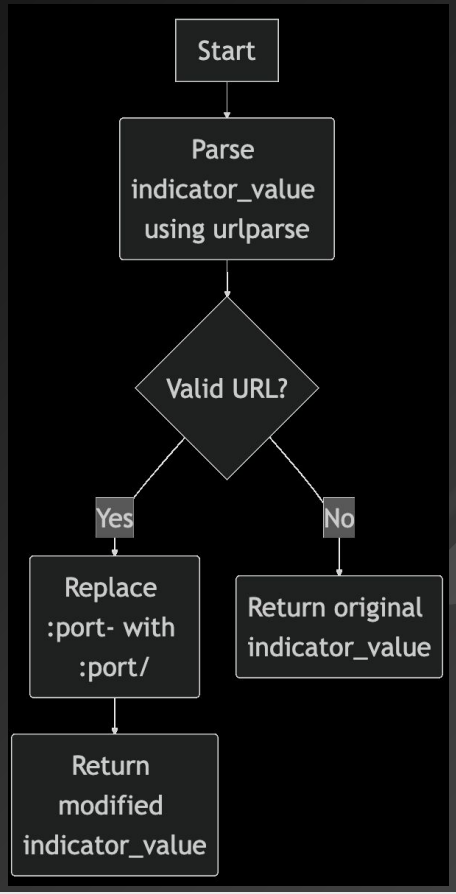
Phase 1: Fanging Indicators - Fang Defanged Indicators



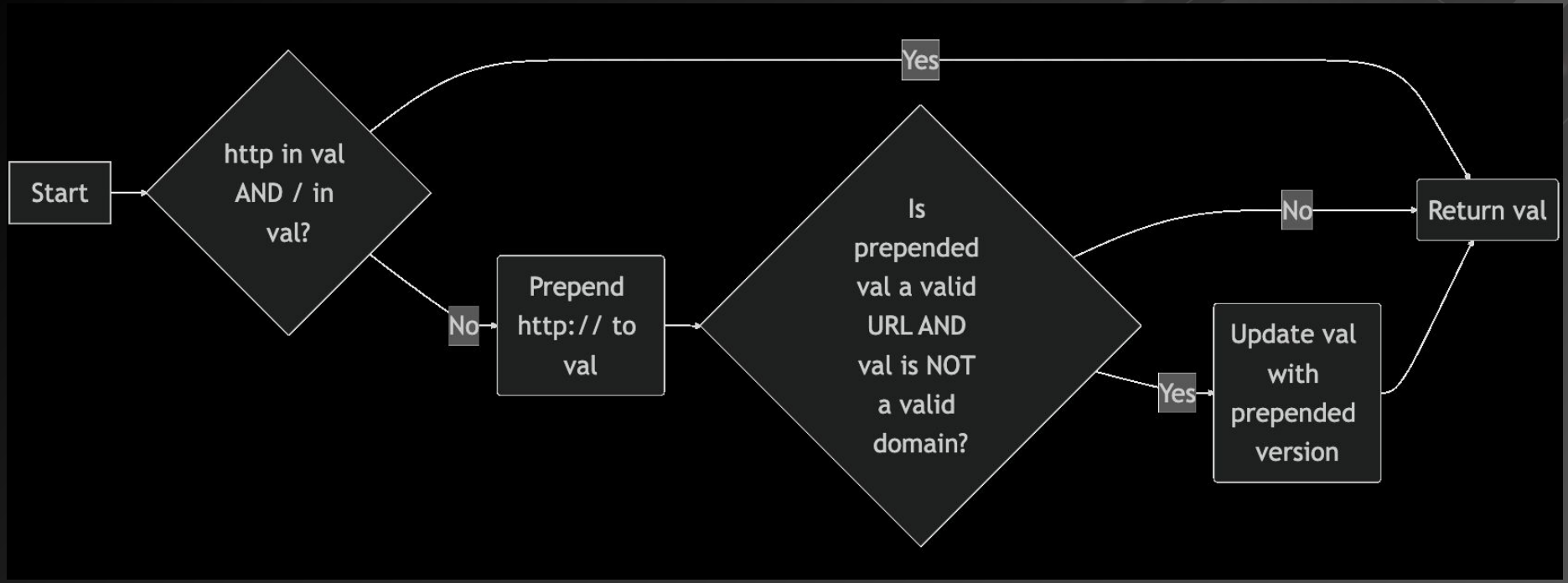
Phase 1: Fanging Indicators - Check and Convert IPs with Ports



Phase 1: Fanging Indicators - Adjust URL Port Validation



Phase 1: Fanging Indicators - Add HTTP to Valid URLs



Phase 1: Summary

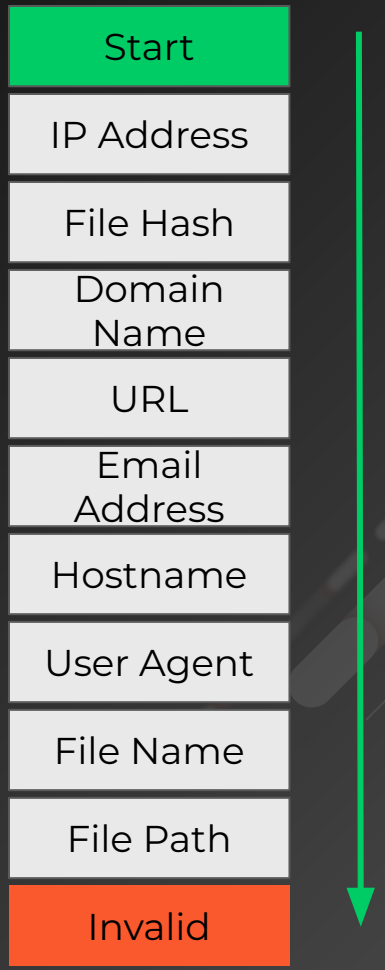
1. Replace all Problematic and Placeholder Characters/Text
2. Convert IPs with Ports to URL formatting
3. Adjust for the URLParse edge cases with ports followed by ‘-’
4. Add the scheme (http) to URLs that do not have it

Phase 2: Identifying IOC Type for Data Quality Purposes

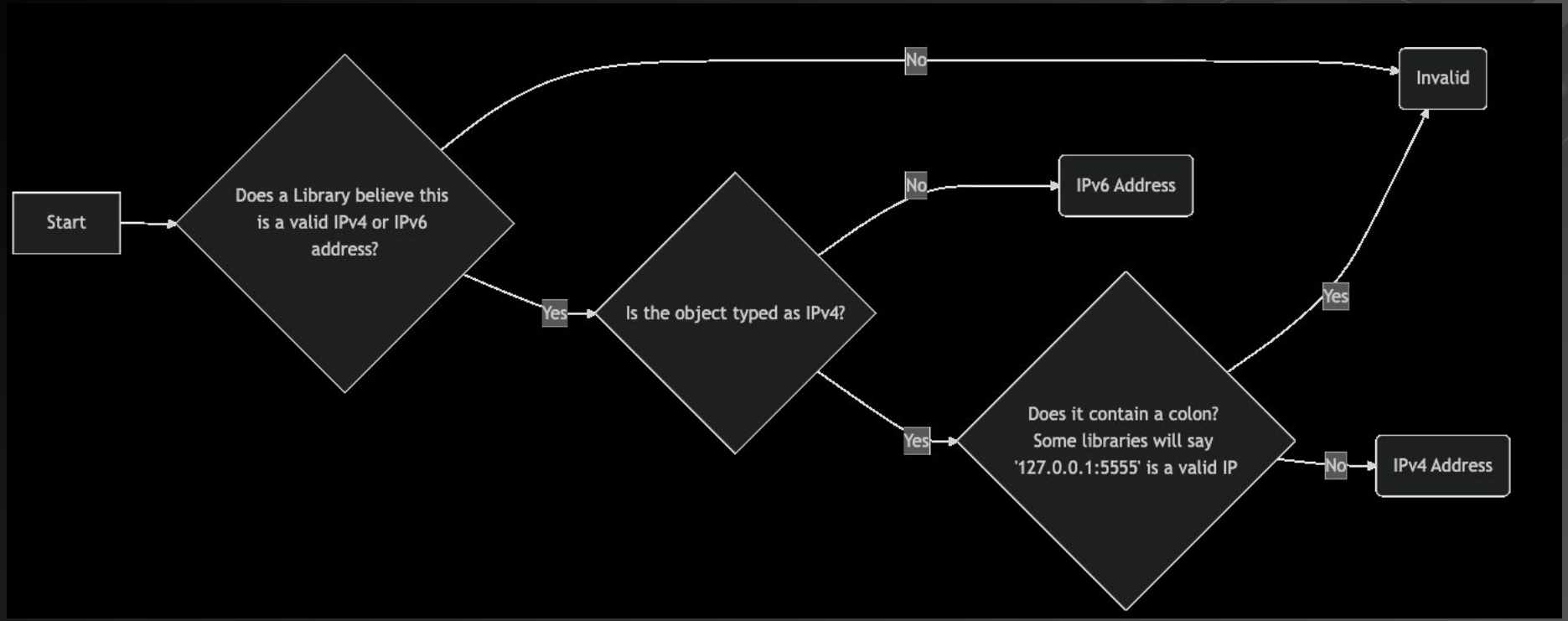
- Now that the data is in a consistent format, the second phase can do its magic
- The secret to parsing IOC type as accurately as possible is to start with the types of IOCs that have the **most consistent patterns** and progress into the **least consistent IOC patterns**
- In my data, it ended up being the following order:

 |

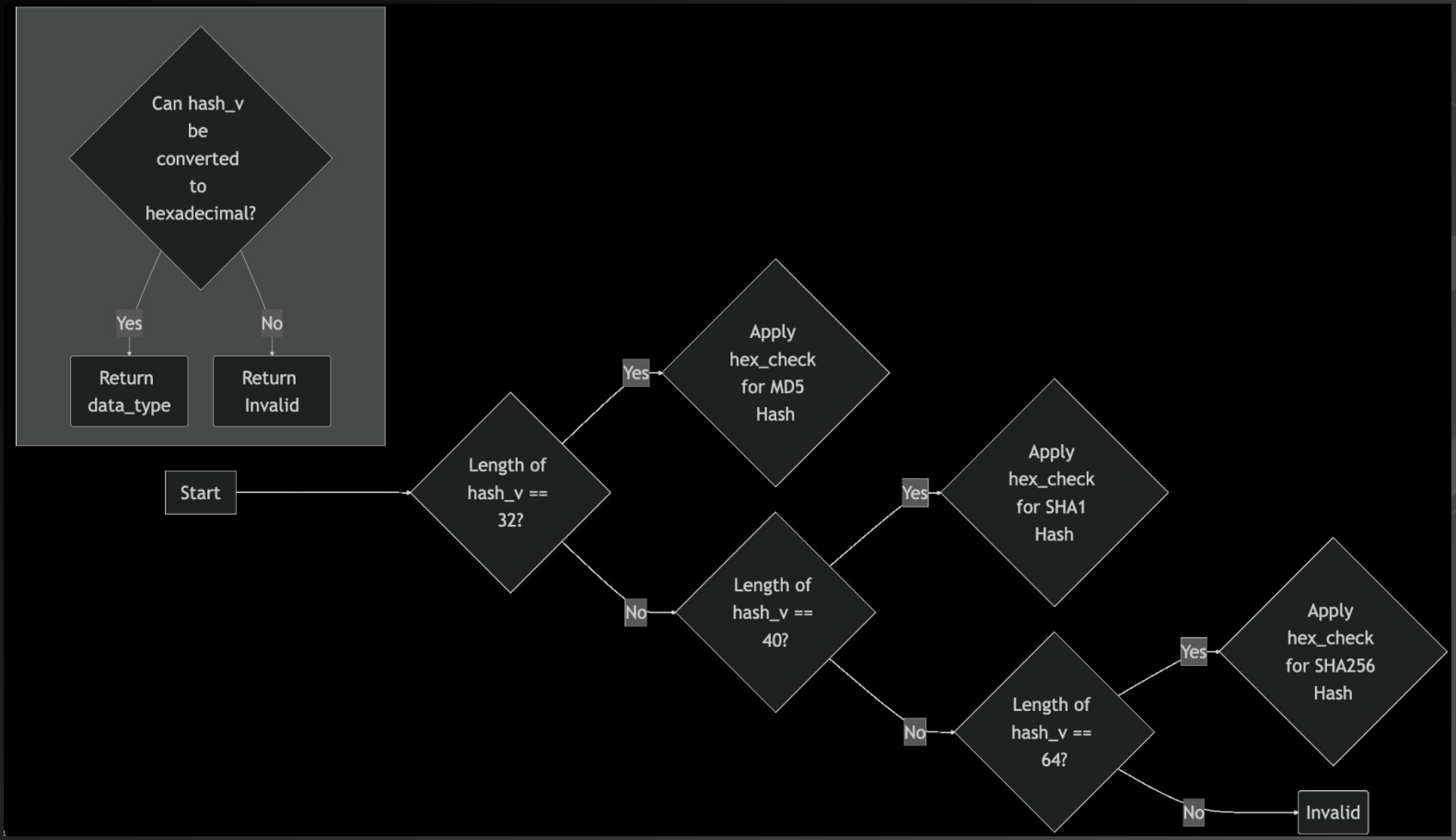
Phase 2:
The IOC Order of Operations



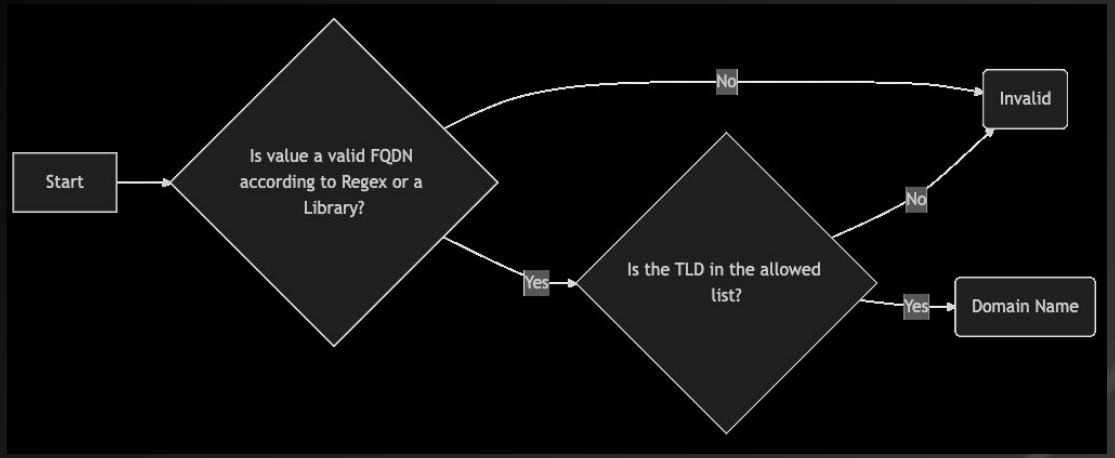
Phase 2: Identifying and Parsing IP Addresses



Phase 2: Identifying and Parsing File Hashes

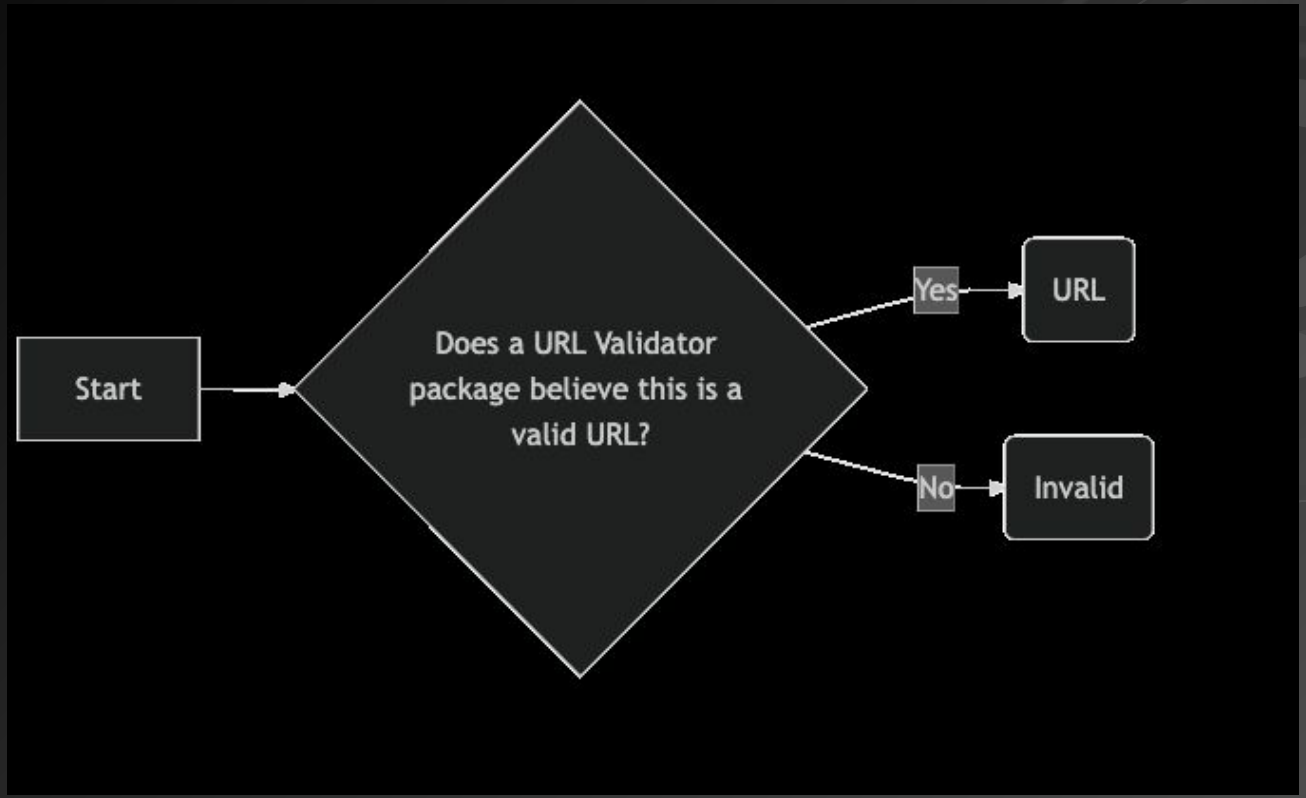


Phase 2: Identifying and Parsing Domain Names

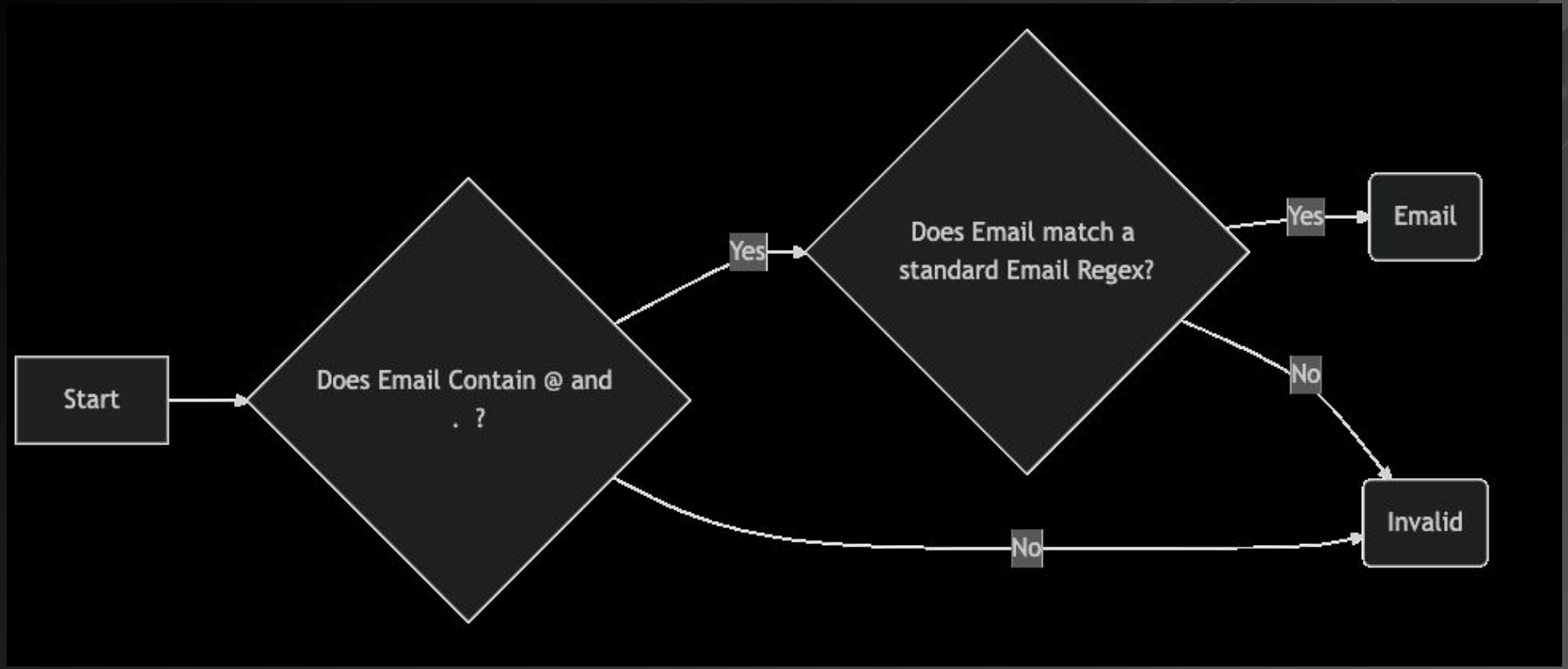


TLDs that are also used for executable types are problematic and may need to be excluded or otherwise manually reviewed.

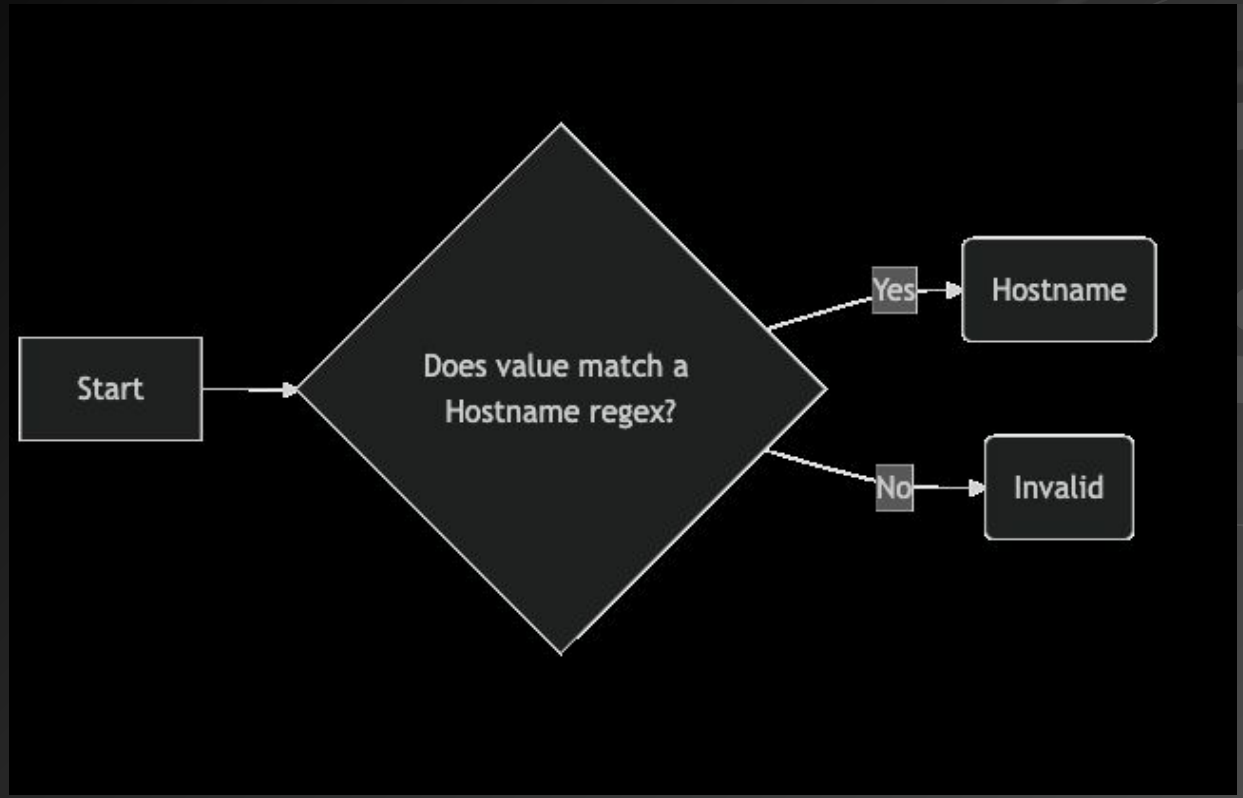
Phase 2: Identifying and Parsing URLs



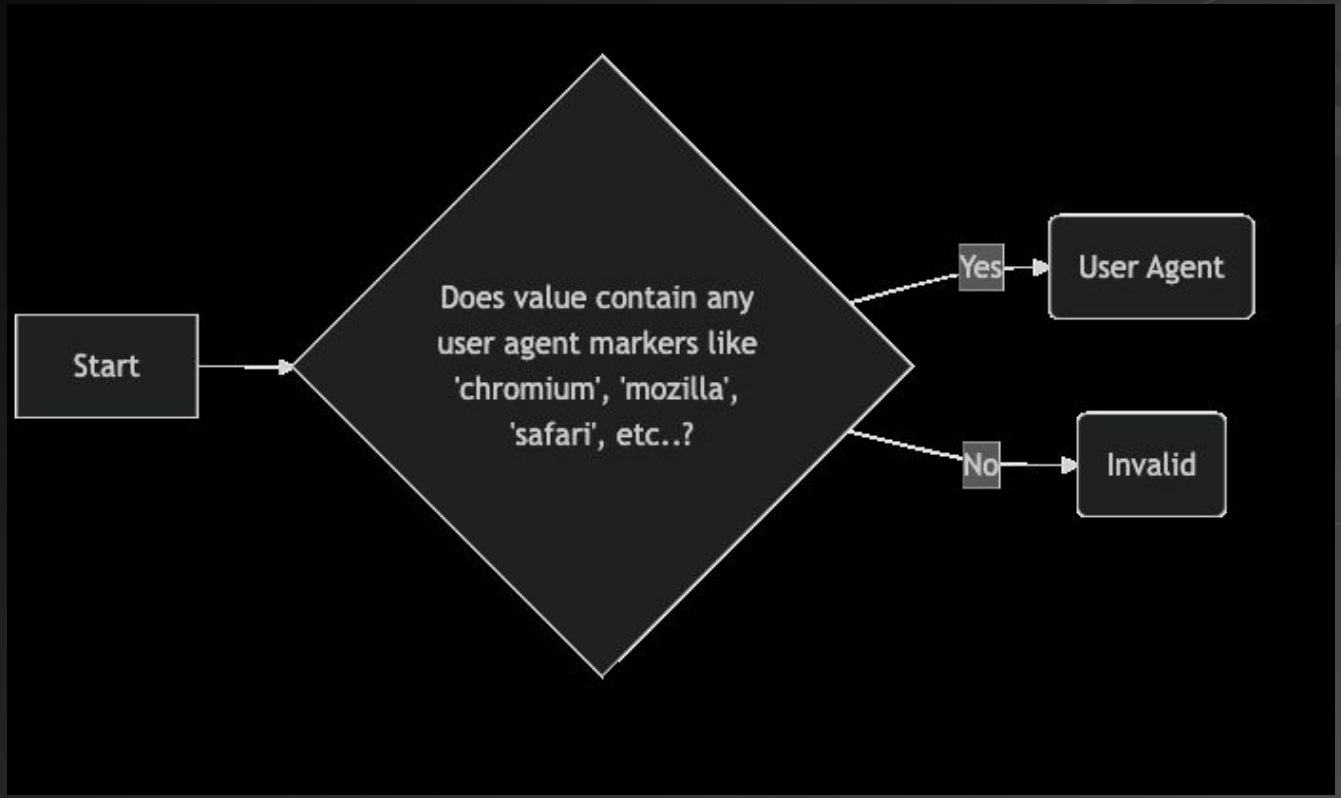
Phase 2: Identifying and Parsing Email Addresses



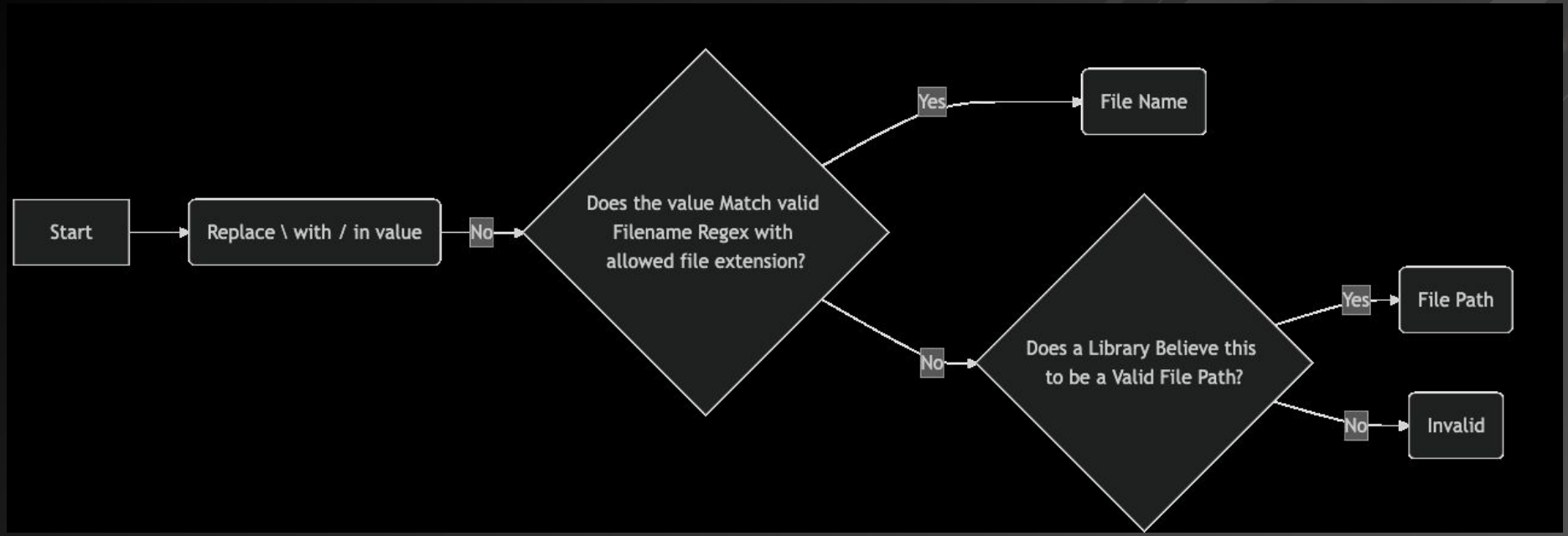
Phase 2: Identifying and Parsing Hostnames



Phase 2: Identifying and Parsing User Agents



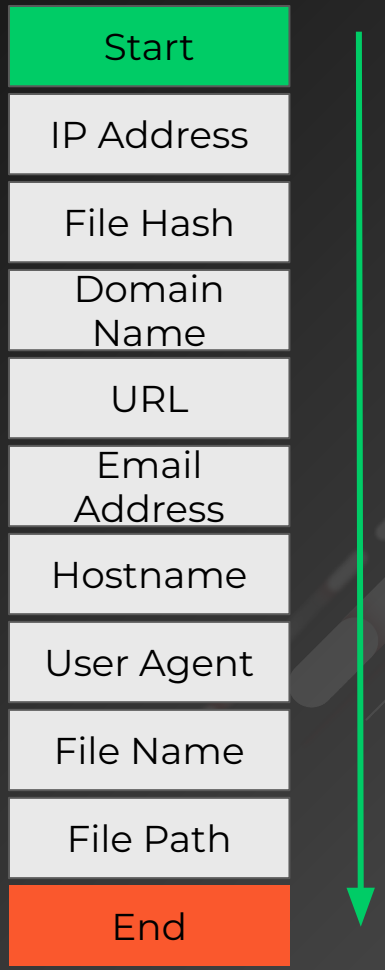
Phase 2: Identifying and Parsing File Names and File Paths



Phase 2: Logging Indicators That Don’t Match Anything

```
if ioc_obj.s_type == "Invalid":
    logger.warning(
        f"Unable to Process Indicator: {ioc_obj.i_value}, listed as Type: {ioc_obj.i_type}, Notes: {ioc_obj.i_notes}"
    )
    return ["Unable"]
```

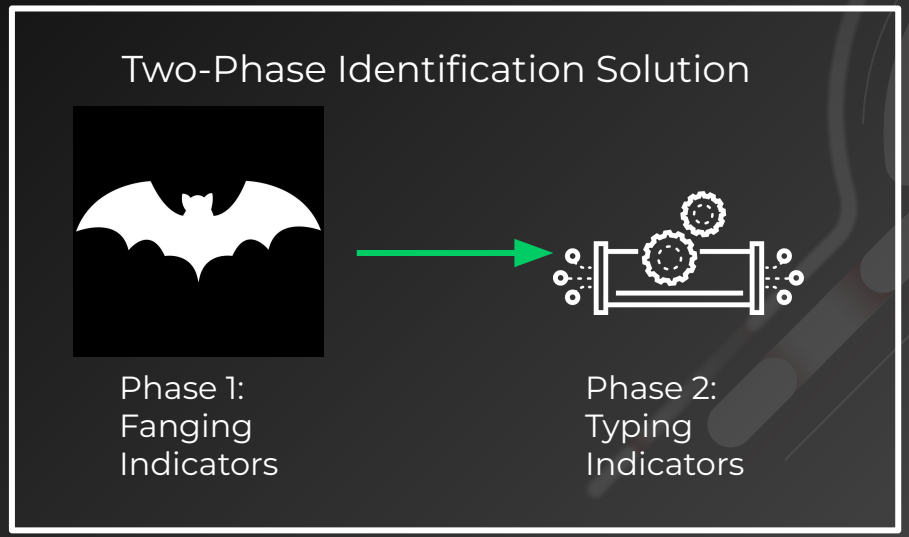

Phase 2:
The IOC Order of Operations



IOC Order of Operations Recap

- Create the ‘sieve’ to catch the biggest (easiest to identify) IOCs at the top, and the smallest (hardest to identify) at the bottom
- Curtail your parsing & identification rules to meet your dataset
- Log unknown indicators for analysis
- Determine which ‘Types’ make sense for your use case
- At this point, you can export to your data analysis/store of choice!

Two Phase Solution Overview



Phase 1 and Phase 2 Recap (cont.)

1. Fang all indicators by reversing known defanging patterns, prepare the data on the back end for identification
2. Run all indicators through the IOC Order of Operations, providing a type or ‘Unknown’ for every Indicator

Results Before and After This Implementation

- For my data, our group went from a **47.8%** capture rate to **99.8%** capture rate (**52% increase**)
- We increased our number of IOCs as well, yielding a **410%** increase in total IOCs identified.
- We have not had more than a few false positives reported by threat intelligence, and those have been quickly remedied due to the piecemeal nature of the IOC order of operations

Future Optimizations

- Identify more high-demand IOCs with Threat Intelligence
- Determine more complicated parsing rules for more complex IOCs
- Eventually, maybe support identifying entire script text (Powershell, Bash, etc.)

Ways This Approach Can Be Modified To Fit Your Use Case

- Input and Output are completely arbitrary, fit them to your needs with a wrapper
- Add/Subtract IOC types to fit your particular data and needs

Lessons Learned

- Don’t trust third-party libraries by default. They often do not account for edge cases
- Avoid ingesting defanged IOCs: they break a lot of threat intelligence tools that depend on defanging rules
- Work in tandem with Threat Intelligence to gain an understanding of their needs and specific common patterns they see
- Never make assumptions about data entry
- Avoid pulling all nighters to prototype tools that need a few months of work and maturation to get concise and accurate data
- Work on cool projects so your company is willing to send you to Ireland for a week

UNIT 42
BY PALO ALTO NETWORKS

Thank you!

Questions?