Patch Intelligence

Project Outline

22nd April 2023

OVERVIEW

The objective of this research project is to collect, correlate and organize data to create a **Patch Intelligence** Information System.

PROBLEM STATEMENT

Modern day Patch Management has two prominent issues:

- Disconnect in information between vulnerabilities and patches.
    - This leads to an overhead in Vulnerability Management, wherein conveying the **actions** that need to be taken by the IT/Development becomes a challenge.
- Lack of critical ITSM information on patches such as Known Issues, Failure rate and Crash data.
    - A **confidence score** in patches can help IT/Development teams take decisions on patch application and ensure the infrastructure is not operationally impacted.
- Cataloging of available mitigation and workaround information
    - If patches can't be applied due to known issues, how to **stay secure**?

Building a Patch Intelligence solution that solves the above problems can vastly improve Vulnerability and Patch Management programs for organizations worldwide.

GOALS

1. Collect patch information for top application libraries.
2. Correlate patch information with CPEs (Common Product Enumeration) and CVEs (Common Vulnerability Enumeration).

3. Build an n-n Knowledge Graph for **CVEs -> CPEs -> Patch-Intel**.

SPECIFICATIONS

Securin's Vulnerability Intelligence currently holds correlated data on CVEs and CPEs. Majority of the information sources on CVEs, chiefly NVD (National Vulnerability Database) and MITRE fail to collate Patch Data. NVD's references or patch links mostly point to advisories rather than actual patches.

For a few vendors, the Securin VI has already captured the patch information such as Microsoft, Apple, Red Hat etc.

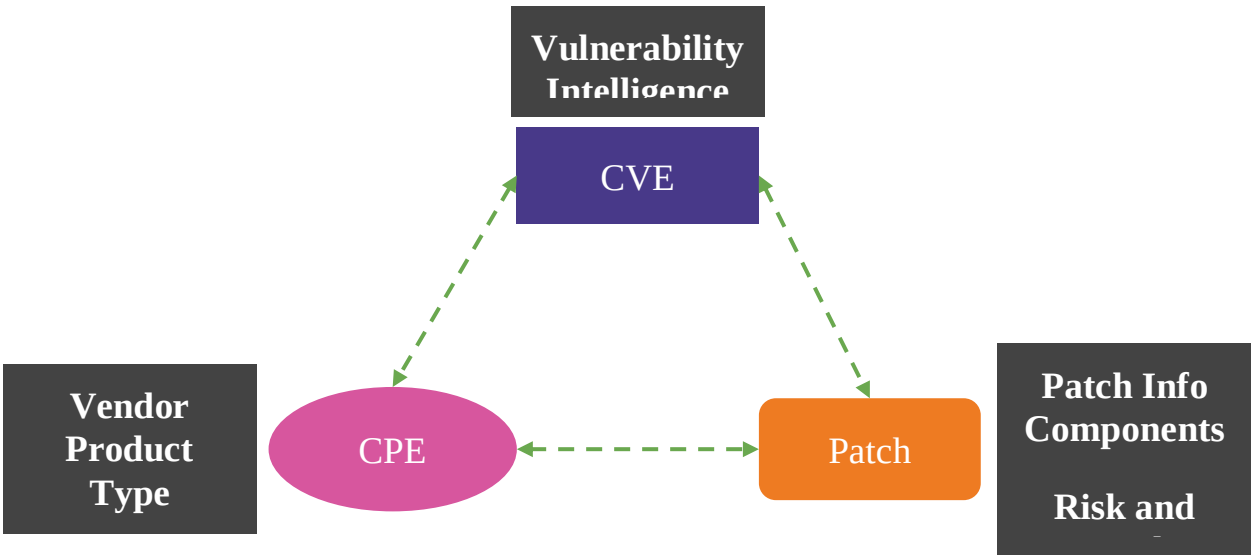
The actionable structure of Patch Information must have the following components:

- **Vendor**
- **Product**
- **Fixed Version**
- **Reference Knowledge Base**
- **Vulnerabilities Fixed**
- **Corresponding NVD CPE**

In addition, for the second use case we are required to gather intelligence from vendor disclosures, open source, forums etc. to keep record of various decisive features for the patch. For example,

- **Known Performance Issues**
- **Crash Likelihood**
- **Reboot Requirements**
- **End of Lifecycle Information**
- **And more..**

Thus building a bidirectional knowledge graph as follows:



MILESTONES

Collection

As detailed above, the first initiative will be directed towards collecting the Patch Data, from vendors or known databases, and putting it together in the actionable structure.

The Patch Risk and Crash intelligence components may be available in vendor disclosures or in open source forums.

Correlation

The Patch Data will need definitive relationships to CVEs and CPEs information present in VI Data.

Organization

Build a parsable Knowledge Graph as depicted in the specifications.

Automation

Create automations to support periodic updation of the data.

DELIVERABLES

Application Libraries Graph Database

Deliver a Graph Database for information collected on the following package libraries

- npm
- Maven
- PyPi
- Linux
- NuGet

The components of the database must include:

- Name
- Environment
- Version
- Security Issues
    - Impacted & Fixed Versions
- Bugs/Caveats/Performance Issues
    - Impacted Versions & Fixed Versions
    - Severity of operational impact (if available)
- Metadata

Knowledge Graph Implementation

Knowledge graph by associating data collected with known vulnerabilities (CVEs) or non-CVE security issues, and create CVE-CPE-Patch relationships as specified above.

For every package version node, the graph must at the minimum provide connected nodes to:

- Security Issues (CVE/Non-CVE) it is vulnerable to.
- Security Issues (CVE/Non-CVE) fixed in this version.
- Bugs/Caveats introduced in this version.
- Bugs/Caveats fixed in this version.
- Supersedence information.

Sustainable Automation

Automate the process for sustainability & repeatability from data collection to knowledge graph updation. The implementation must be aligned with the requirements set by the Securin Product team.

- Python-based implementation.
- Graph DB used must support API functionalities.

Technical Documentation

Documentation providing in depth details on concept, technicals & codebase in the format provided by Securin.

KNOWLEDGE & SKILLS REQUIRED

Research & Analytics

Research web-wide available sources for:

- Data collection
- Pattern Recognition
- Enumerating statistical analyses

Graphs in Data Structure

Understanding of Graphs basics -

- Graph Terminology
- Directed and Undirected Graphs
- Parent:Child Relationships
- Graphs Implementation as Data Structures
    - ArangoDB/Neptune

Python Basics

Basic Python web-scraping and data structure implementation skills.

- BeautifulSoup
- Pandas

Information Technology Basics

Understanding of Information Technology basics such as operating systems, application softwares and IT infrastructure.

Information Security Basics

Understanding of Information Security basics such as vulnerabilities, impact by vendors-products, patches and remediation activities.