Team 1

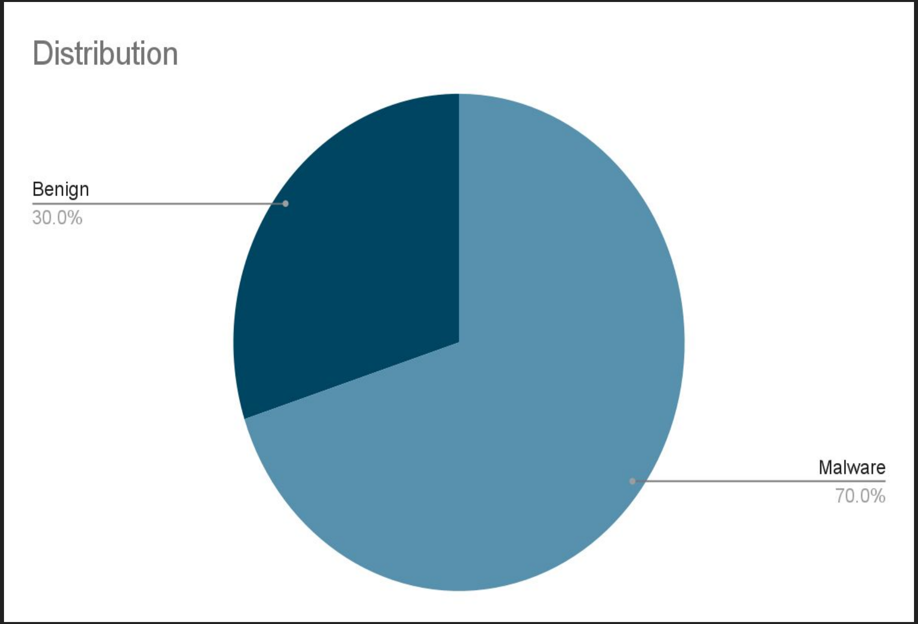
Team Members
Ayushri, Akshat, Akshat

Dataset

PE header dataset

- 138,000 executable files: .exe / .dll
- Imbalanced towards malware files
    - Malware: 96.7K
    - Goodware: 41.3K

- 57 Features in the original dataset.
- Features used: 24



Source: https://www.kaggle.com/datasets/dasarijayanth/pe-header-data/data

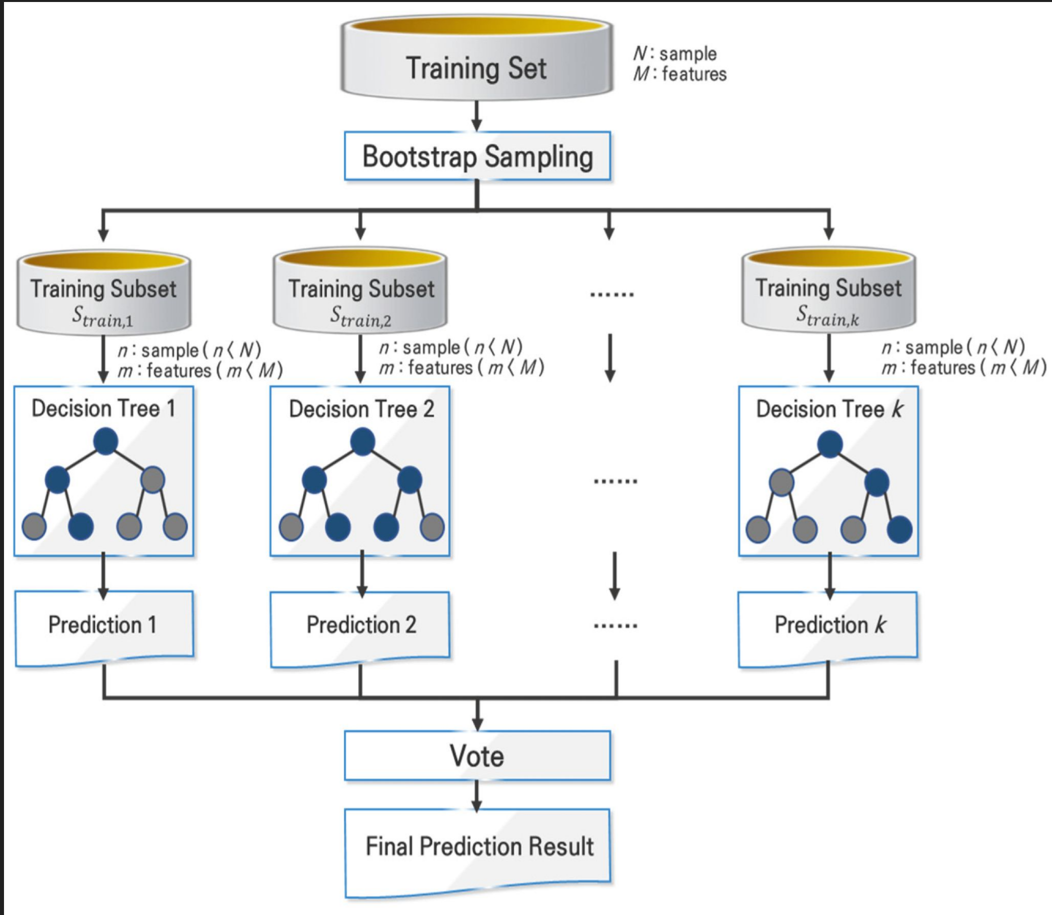
Features

- Characteristics
- MajorLinkerVersion
- MinorLinkerVersion
- SizeOfCode
- SizeOfInitializedData
- SizeOfUninitializedData
- AddressOfEntryPoint
- BaseOfCode
- BaseOfData
- SectionsNb
- SectionsMeanEntropy
- ImportsNb
- ExportNb
- ImageBase
- SectionAlignment
- FileAlignment
- MajorOperatingSystemVersion
- MinorOperatingSystemVersion
- MajorImageVersion
- MinorImageVersion
- MajorSubsystemVersion
- MinorSubsystemVersion
- SizeOfHeaders
- SizeOfHeapCommit

Model Selection

Model Architecture: Random Forest

- Number of estimators: 100
- Train-validation Split: 80-20
- Threshold: 0.6
- Criterion: Gini Index



```python
def calculate_mean_entropy(sections):
    entropies = [section.entropy for section in sections]
    return sum(entropies) / len(entropies) if entropies else 0

def extract_file_info(file_content):
    """Extract information from a PE file using lief."""
    with tempfile.NamedTemporaryFile(delete=False) as temp_file:
        temp_file.write(file_content)
        temp_file_path = temp_file.name

    info = []
    binary = lief.PE.parse(temp_file_path)
    info.append(binary.optional_header.sizeof_headers)
    info.append(binary.header.characteristics)
    info.append(binary.optional_header.major_linker_version)
    info.append(binary.optional_header.minor_linker_version)
    info.append(binary.optional_header.sizeof_code)
    info.append(binary.optional_header.sizeof_initialized_data)
    info.append(binary.optional_header.sizeof_uninitialized_data)
    info.append(binary.optional_header.addressof_entrypoint)
    info.append(binary.optional_header.baseof_code)
    info.append(binary.optional_header.baseof_data if hasattr(binary.optional_header, 'baseof_data') else 0)
    info.append(binary.optional_header.imagebase)
    info.append(binary.optional_header.section_alignment)
    info.append(binary.optional_header.file_alignment)
    info.append(binary.optional_header.major_operating_system_version)

    info.append(calculate_mean_entropy(binary.sections))
    info.append(len(binary.imports))

    os.unlink(temp_file_path)

    return info
```

After BlackBox testing results

Realisations:

- Model might be too biased towards Malware (Aggressive)
- Threshold fine-tuning might be required.
- Experimenting with data-scaling could help

What actually worked:

- Added around all the goodware samples from the Dike Dataset.
- Increasing threshold to 0.68 for best results

Way~~s~~ to bypass

- Try a lot of goodware strings (didn’t work for us but you can try using benign files of Dike dataset)
- Use a dropper (Prof’s dropper is best)

What else we could have done?

- Increased number of estimators for RF
- Different machine learning algorithm
- Pool of models maybe?

Team 1 - Attack

Team Members
Ayushri, Akshat, Akshat

Approaches

- Used Any.Run, HybridAnalysis, VirusTotal
- Too much of information, most of them trojan
- Adding NOPs
- Much difficult to do since one instruction change leads to changes in multiple places
- Used UPX, searched for many packers
- Packed calc.exe detected as malicious
- Appending strings to file
- Easy to do but detected malicious on VirusTotal
- Using dropper
- Found various droppers on github, simple XOR droppers, professor's dropper, difficult to use

Problems faced with Dropper

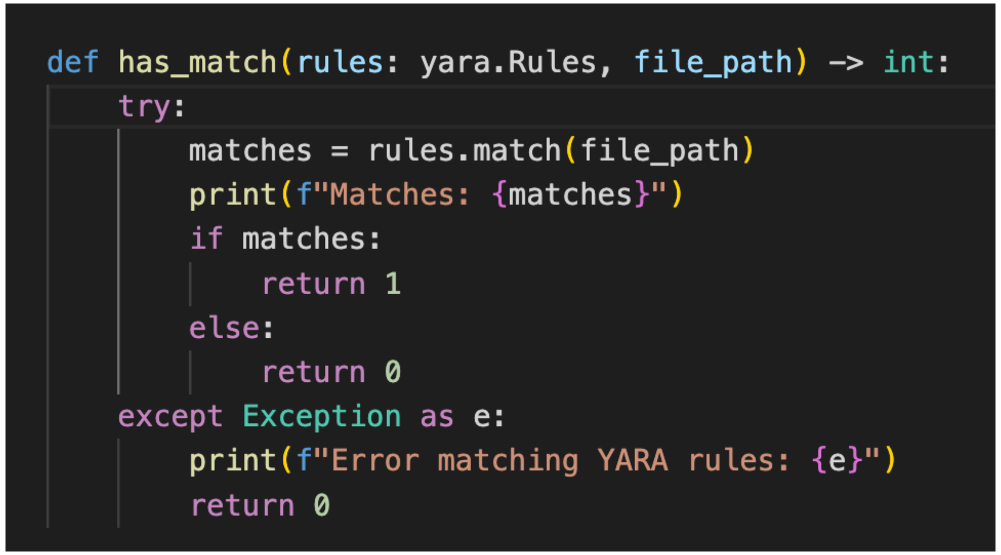
- Difficult to understand (C++)
- Unable to use XOR encoding
- Visual Studio Code extensions didn’t work
- Visual Studio didn’t work in macOS
- Used Windows VM, worked for a while then started complaining about M1
- Borrowed friend’s Windows (with a promise of not infecting it, hopefully)

Black Box Attack Submission

- XORed files with 0x24 (Python)
- Added lot of dead code in dropper
- Created exe files in Visual Studio (1 by 1)
- Submitted on VirusTotal, detection rate reduced!
- Able to bypass some models

White Box Attack Submission

- Team 2 and Team 4 models – difficult to bypass
- Appending goodware strings (a lot) – Team 2 bypassed
- Removed dead code from dropper – some files undetected for Team 4
- Updated code to see which signatures are detected



```python
def has_match(rules: yara.Rules, file_path) -> int:
    try:
        matches = rules.match(file_path)
        print(f"Matches: {matches}")
        if matches:
            return 1
        else:
            return 0
    except Exception as e:
        print(f"Error matching YARA rules: {e}")
        return 0
```

- SIGNATURE_BASE_SUSP_Xored_URL_In_EXE: make file size more than 2mb
- SIGNATURE_BASE_SUSP_Xored_MSDOS_Stub_Message: append "SophosClean" to file

White Box Attack Submission

- ~~XORed files with 0x24 (Python)~~
- ~~Created exe files in Visual Studio (1 by 1)~~
- Created python script to automate :
    1. XOR a file (0x31)
    2. Attach resource (malware file)
    3. Create dropper files
    4. Append good strings
- Used bash script to test on all models