SCANOSS

# SBOMs and Cryptographic Algorithms

Status and Next Steps

FOSDEM 2025
2025-02-02

Matias D’Aloia
Software Engineer
matias.daloia@scanoss.com

## The Crypto Identification Problem

Imagine if crypto algorithms were declared using arbitrary formats – like restaurant reviews.

⭐⭐⭐
⭐⭐

```
"AES-256 in use"
```

9.2/10

```
{"algorithm": "AES", "keySize": 256}
```

👍👍👍👎

```
<crypto alg="AES256" />
```

“Excellent”

```
# AES-256 encryption
encrypted_message = AES.new(os.urandom(32), AES.MODE_EAX).encrypt("Some Encrypted Message")
```

## Why Standarized Crypto Identification Matters

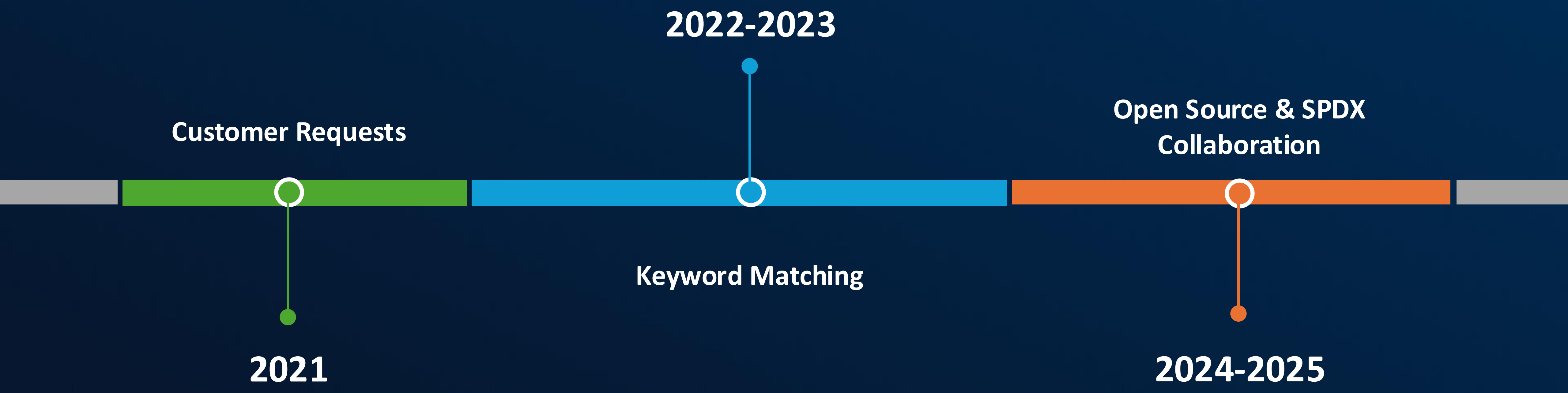
- **To our Key Stakeholders:**
    - Trade compliance teams need accurate ECCN classifications for Export Control
    - Security teams must adhere to NIST standards on CAVP (Crypto Algorithm Validation Program)
    - Companies are increasingly concerned about PQC (Post Quantum Cryptography)
    - Auditing requirements continue to grow

- **For the entire community and to SCANOSS:**
    - Standarization SAVES EFFORT and RESOURCES
    - Standarization boosts collaboration

## An Open Crypto Algorithm Dataset

- Crypto algorithm definition list
- Simple data structure and attributes (proto-taxonomy)
- Machine-Readable format, so extensible
- Reference code for algorithms' list detection
- Battle-tested in production

## Our journey: …towards Open Data



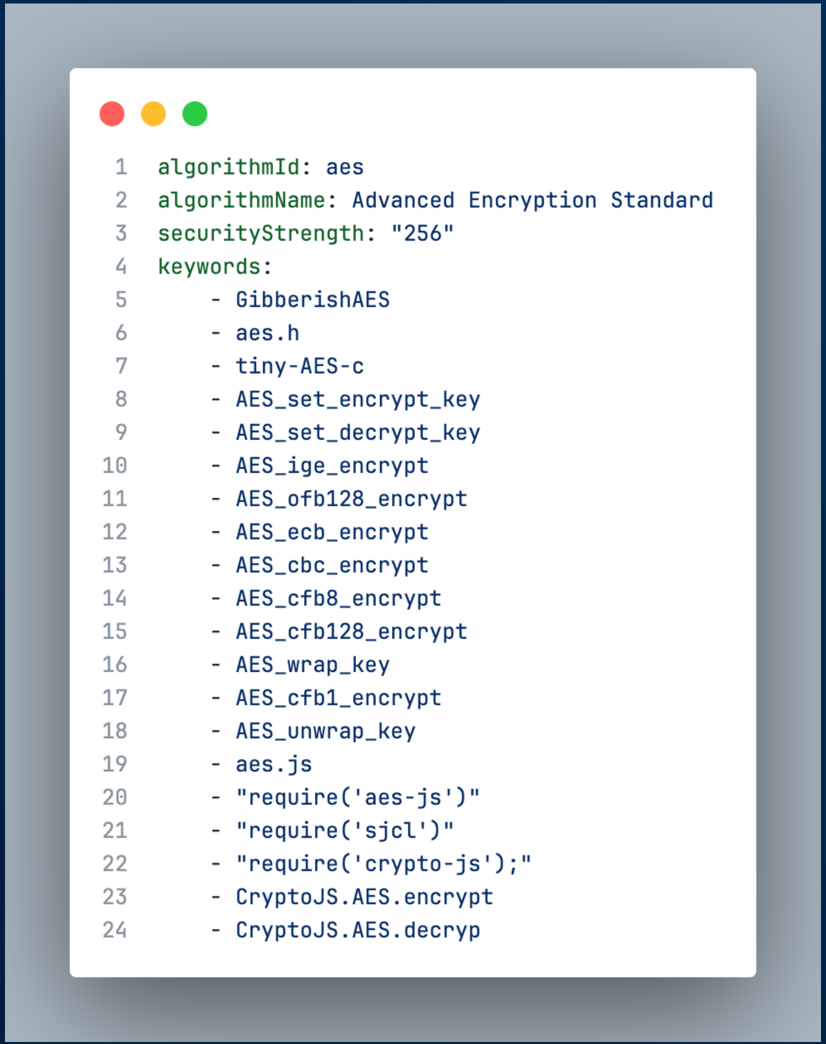
## Driven by Customers Needs

- Customers frequently asked: 'Can you tell us which crypto algorithms are in this open-source project?'
- This wasn't just an internal need; it was a recurring request from real customers.
- We recognized the importance of addressing this critical need for our customers.

## Keyword Matching: A Practical Start

🧠 Sometimes, simple is the smart way!

- One definition file per crypto algorithm
- Effective for large-scale scanning
- Allowed us to be precise on the detection



```yaml
algorithmId: aes
algorithmName: Advanced Encryption Standard
securityStrength: "256"
keywords:
      - GibberishAES
      - aes.h
      - tiny-AES-c
      - AES_set_encrypt_key
      - AES_set_decrypt_key
      - AES_ige_encrypt
      - AES_ofb128_encrypt
      - AES_ecb_encrypt
      - AES_cbc_encrypt
      - AES_cfb8_encrypt
      - AES_cfb128_encrypt
      - AES_wrap_key
      - AES_cfb1_encrypt
      - AES_unwrap_key
      - aes.js
      - "require('aes-js')"
      - "require('sjcl')"
      - "require('crypto-js');"
      - CryptoJS.AES.encrypt
      - CryptoJS.AES.decryp
```

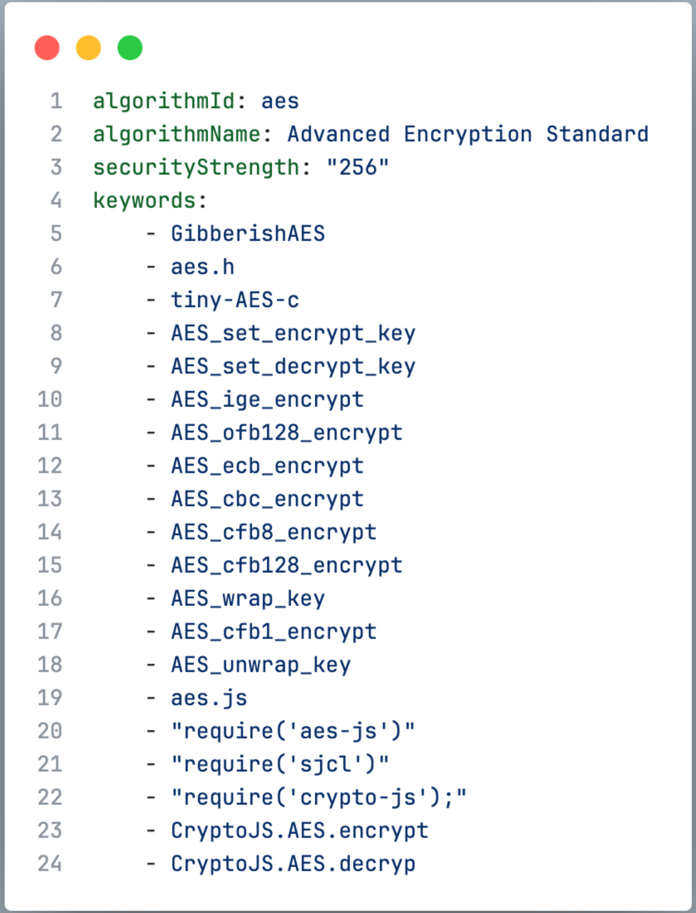
## Keyword Matching: A Practical Start

Sometimes, simple is the smart way!

- One definition file per crypto algorithm
- Effective for large-scale scanning
- Allowed us to be precise on the detection

We realized:

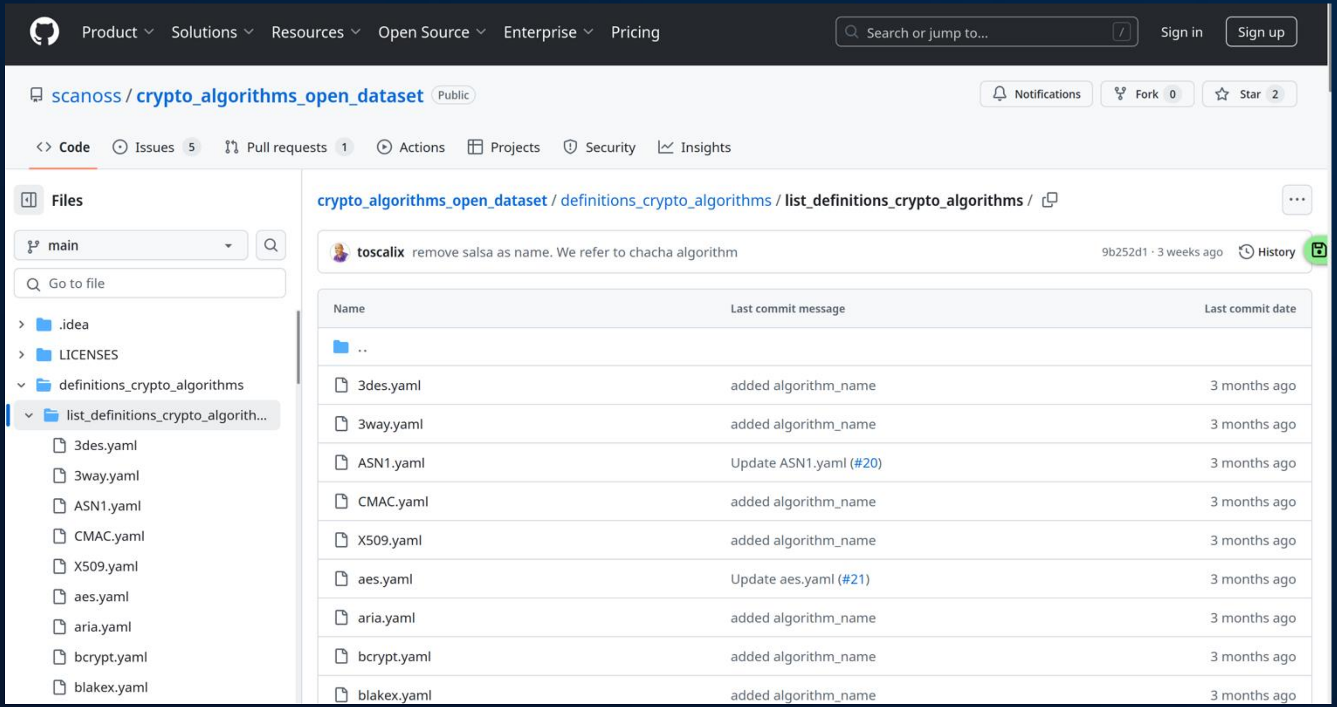
- What about non open-source projects?
- New crypto libraries and frameworks
- The community was already involved



```
algorithmId: aes
algorithmName: Advanced Encryption Standard
securityStrength: "256"
keywords:
    - GibberishAES
    - aes.h
    - tiny-AES-c
    - AES_set_encrypt_key
    - AES_set_decrypt_key
    - AES_ige_encrypt
    - AES_ofb128_encrypt
    - AES_ecb_encrypt
    - AES_cbc_encrypt
    - AES_cfb8_encrypt
    - AES_cfb128_encrypt
    - AES_wrap_key
    - AES_cfb1_encrypt
    - AES_unwrap_key
    - aes.js
    - "require('aes-js')"
    - "require('sjcl')"
    - "require('crypto-js');"
    - CryptoJS.AES.encrypt
    - CryptoJS.AES.decryp
```

## Open to the World: Standardization and Community

- Released the dataset under CC0 License
- From de-facto standard → To SPDX collaboration



scanoss / crypto_algorithms_open_dataset Public

crypto_algorithms_open_dataset / definitions_crypto_algorithms / list_definitions_crypto_algorithms /

.idea
LICENSES
definitions_crypto_algorithms
list_definitions_crypto_algorith...
3des.yaml
3way.yaml
ASN1.yaml
CMAC.yaml
X509.yaml
aes.yaml
aria.yaml
bcrypt.yaml
blakex.yaml

toscalix remove salsa as name. We refer to chacha algorithm 9b252d1 · 3 weeks ago

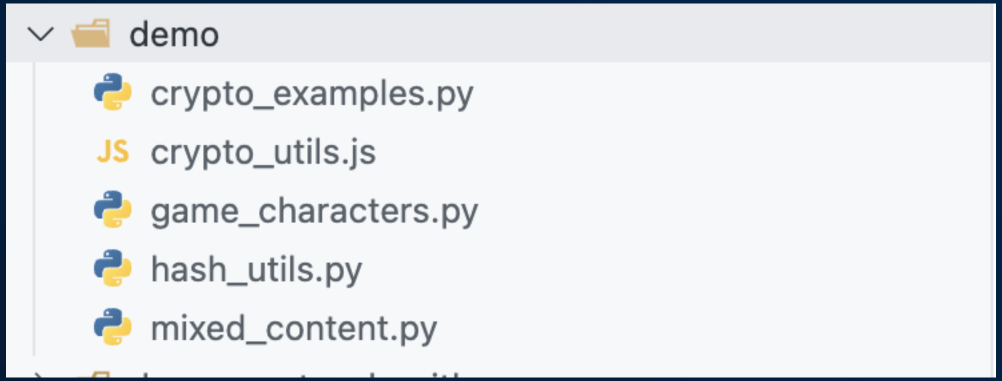
| Name | Last commit message | Last commit date |
|---|---|---|
| .. | | |
| 3des.yaml | added algorithm_name | 3 months ago |
| 3way.yaml | added algorithm_name | 3 months ago |
| ASN1.yaml | Update ASN1.yaml (#20) | 3 months ago |
| CMAC.yaml | added algorithm_name | 3 months ago |
| X509.yaml | added algorithm_name | 3 months ago |
| aes.yaml | Update aes.yaml (#21) | 3 months ago |
| aria.yaml | added algorithm_name | 3 months ago |
| bcrypt.yaml | added algorithm_name | 3 months ago |
| blakex.yaml | added algorithm_name | 3 months ago |

## Seeing it in Action

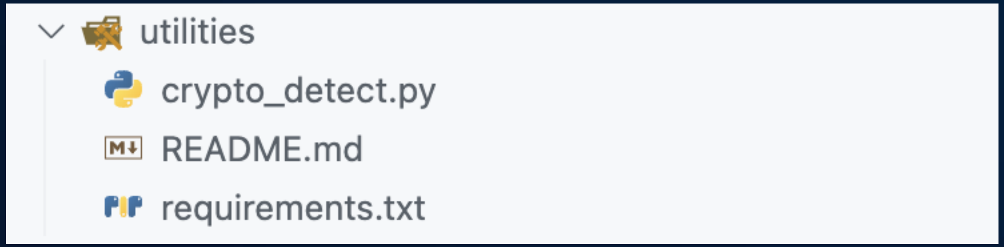
Demo repo branch

EXAMPLE

## Seeing it in Action


demo
crypto_examples.py
crypto_utils.js
game_characters.py
hash_utils.py
mixed_content.py

Files containing crypto algorithms


utilities
crypto_detect.py
README.md
requirements.txt

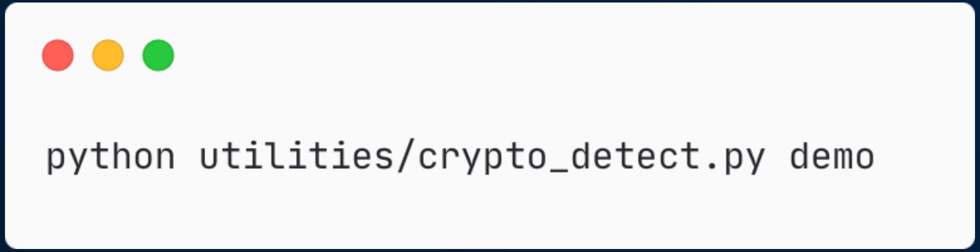
Example script for detection

## Seeing it in Action

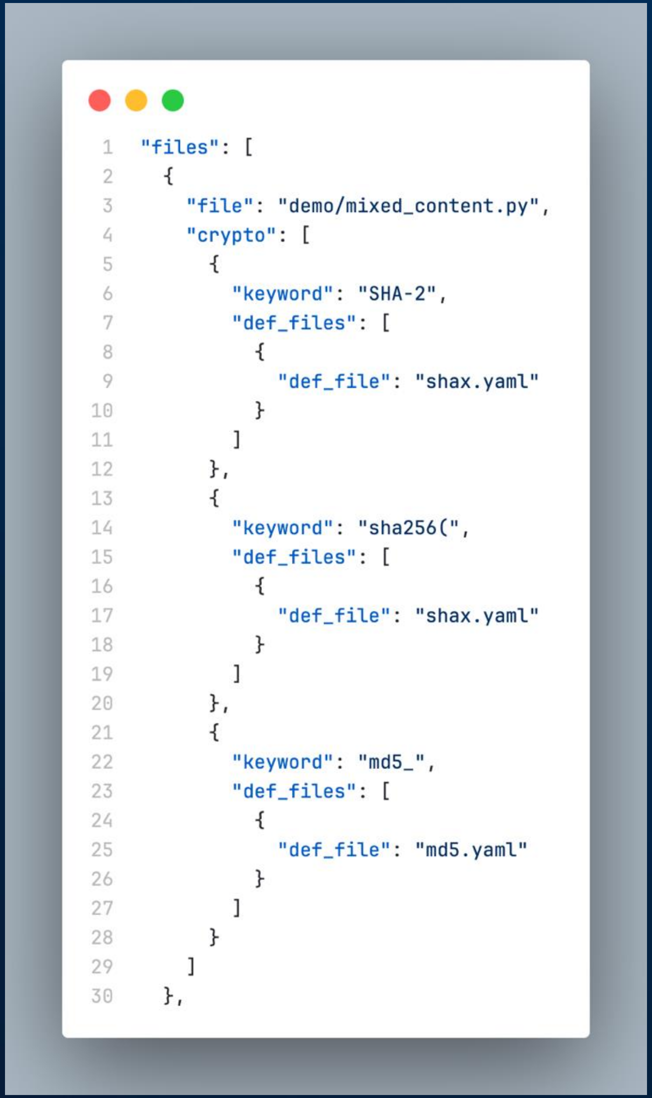
```
python utilities/crypto_detect.py demo
```

EXAMPLE

## Seeing it in Action



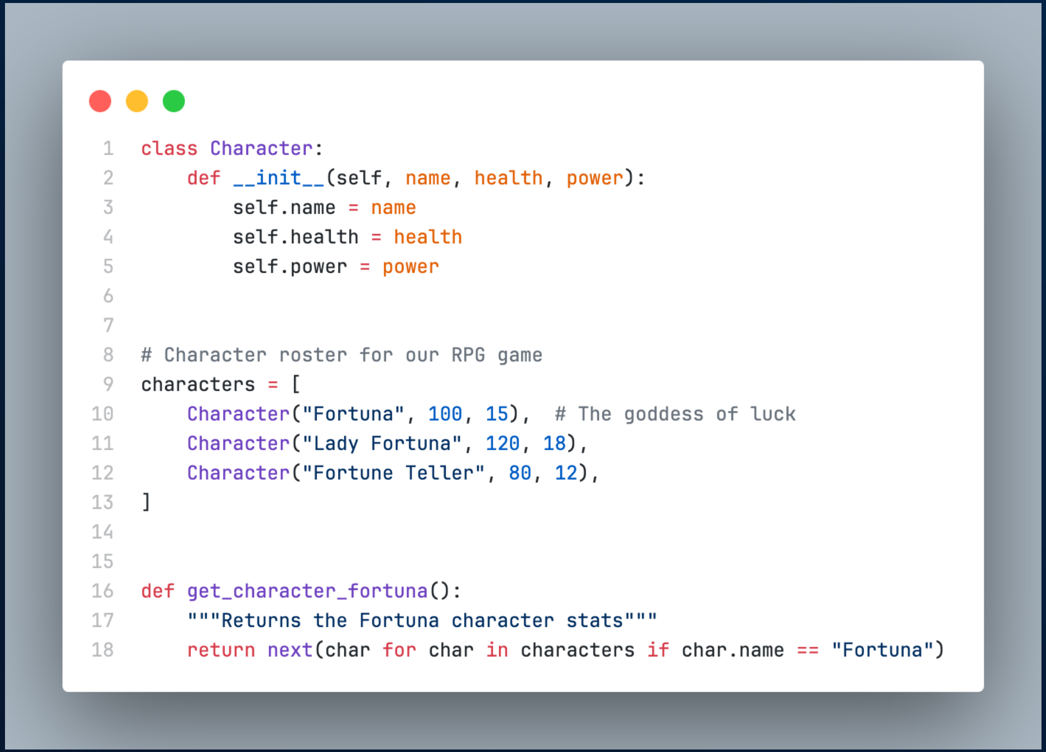
```
python utilities/crypto_detect.py demo
```



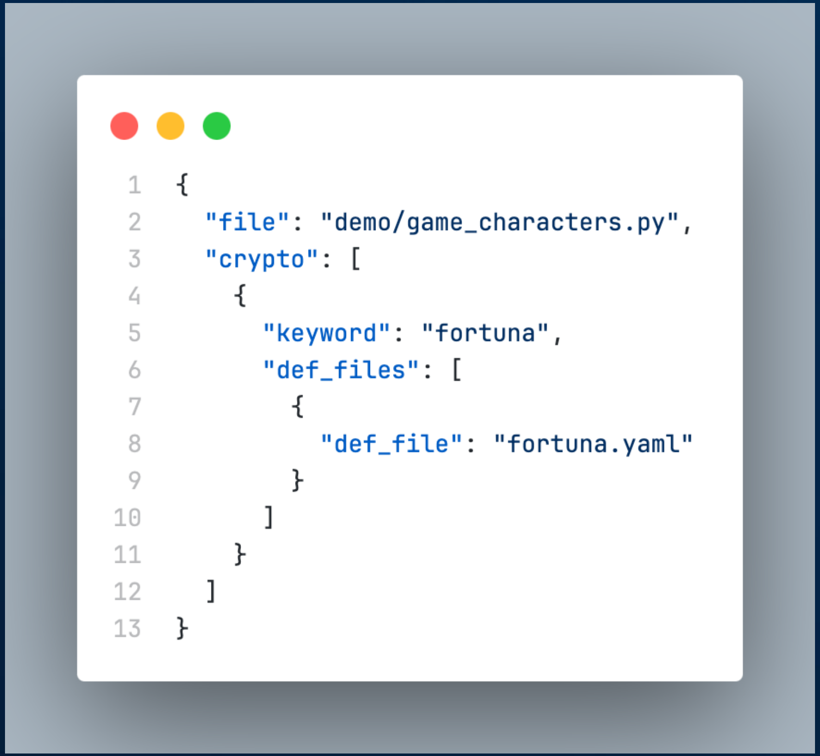
```
"files": [
  {
    "file": "demo/mixed_content.py",
    "crypto": [
      {
        "keyword": "SHA-2",
        "def_files": [
          {
            "def_file": "shax.yaml"
          }
        ]
      },
      {
        "keyword": "sha256(",
        "def_files": [
          {
            "def_file": "shax.yaml"
          }
        ]
      },
      {
        "keyword": "md5_",
        "def_files": [
          {
            "def_file": "md5.yaml"
          }
        ]
      }
    ]
  },
```

## Beyond Keyword Matching: Addressing Context

- Keyword in a different context:



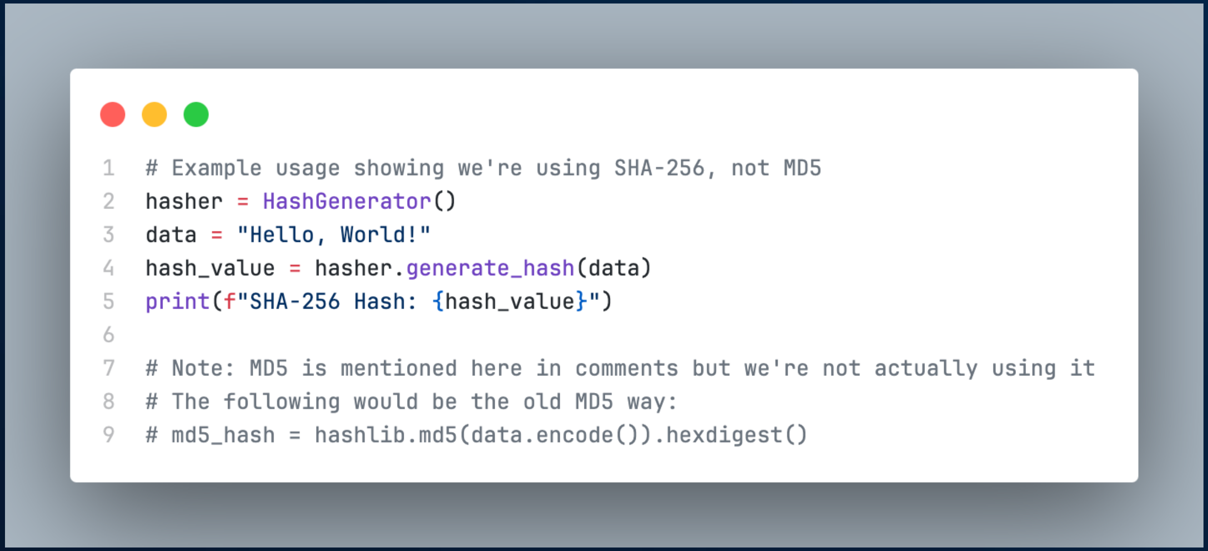
```python
class Character:
    def __init__(self, name, health, power):
        self.name = name
        self.health = health
        self.power = power


# Character roster for our RPG game
characters = [
    Character("Fortuna", 100, 15),  # The goddess of luck
    Character("Lady Fortuna", 120, 18),
    Character("Fortune Teller", 80, 12),
]


def get_character_fortuna():
    """Returns the Fortuna character stats"""
    return next(char for char in characters if char.name == "Fortuna")
```



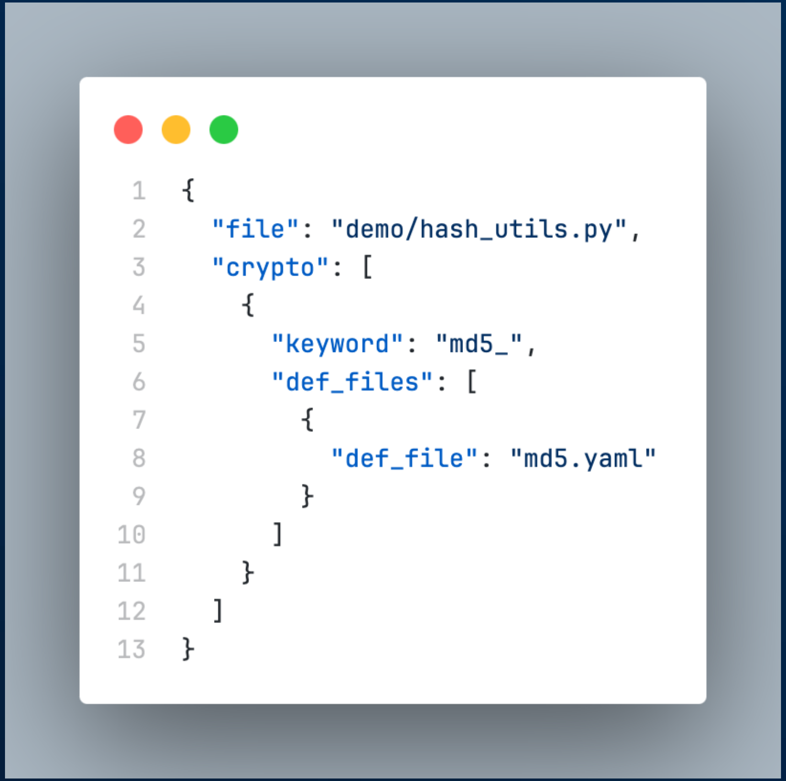
```json
{
  "file": "demo/game_characters.py",
  "crypto": [
    {
      "keyword": "fortuna",
      "def_files": [
        {
          "def_file": "fortuna.yaml"
        }
      ]
    }
  ]
}
```

## Beyond Keyword Matching: Addressing Context

- Misleading comment:



```python
# Example usage showing we're using SHA-256, not MD5
hasher = HashGenerator()
data = "Hello, World!"
hash_value = hasher.generate_hash(data)
print(f"SHA-256 Hash: {hash_value}")

# Note: MD5 is mentioned here in comments but we're not actually using it
# The following would be the old MD5 way:
# md5_hash = hashlib.md5(data.encode()).hexdigest()
```



```json
{
  "file": "demo/hash_utils.py",
  "crypto": [
    {
      "keyword": "md5_",
      "def_files": [
        {
          "def_file": "md5.yaml"
        }
      ]
    }
  ]
}
```

## Looking Ahead

- Software Transparency Foundation
- New Implementations
- Community Growth

## Call For Participation

- Improve the Dataset
- Create New Implementations
- Share Real-World Use Cases
- Explore AI/ML for Context

## Speaking The Same Language

Standardized crypto identification enables better collaboration and a more secure software ecosystem

## Thank You!

- SCANOSS SCA Open-Source Tools: https://github.com/scanoss
- Crypto Algorithms Open Dataset:

  https://github.com/scanoss/crypto_algorithms_open_dataset
- Purl to CPE: https://github.com/scanoss/purl2cpe
- STF Web: https://www.softwaretransparency.org
- osskb.org Web: https://osskb.org