Threat Research Blog

ELFant in the Room – capa v3

September 15, 2021 | by Willi Ballenthin, Moritz Raabe, Mike Hunhoff, Ana Maria Martinez Gomez

REVERSE ENGINEERING MALWARE FLARE

Since our initial public release of capa, incident responders and reverse engineers have used the tool to automatically identify capabilities in Windows executables. With our newest code and ruleset updates, capa v3 also identifies capabilities in Executable and Linkable Format (ELF) files, such as those used on Linux and other Unix-like operating systems. This blog post describes the extended analysis and other improvements. You can download capa v3 standalone binaries from the project's release page and checkout the source code on GitHub.

ELF File Format Support

capa finds capabilities in programs by parsing executable file formats, disassembling code, and then recognizing features in functions. In versions v1 and v2, capa only understood the PE file format, so its analysis was restricted to Windows programs. Thanks to our colleagues at Intezer, capa now recognizes ELF files! This means you can use the tool to identify behaviors in malware that targets Linux computers. Figure 1 shows a rule that describes techniques to fetch the current user on Linux.

```
rule:
  meta:
    name: get current user on Linux
    namespace: collection
    author: joakim@intezer.com
    scope: function
    examples:
      - 7351f8a40c5450557b24622417fc478d:0x405438
  features:
    - and:
      - os: linux
      - or:
        - and:
          - api: geteuid
          - api: getpwuid
        - api: getlogin
        - api: getlogin_r
        - api: cuserid
```

Figure 1: capa rule identifying capabilities on Linux

We're excited Intezer leverages capa and thrilled they are sharing their improvements with the community. In addition to the code updates, Intezer proposed 36 capa rules to identify various capabilities in ELF files, such as reconnaissance, persistence, and host

interaction techniques. Please read Intezer's blog post for more details.

New Features capa Can Recognize

As we taught capa to recognize ELF files, we also wanted rule authors to tune their rules to find behaviors specific to different operating systems (OS), CPU architectures, and file formats. For example, the APIs exposed by Windows are very different from those found on Linux systems; therefore, rules should clearly designate which pattern to use on Windows versus Linux.

Based on discussions and feedback collected from users and contributors, we've extended capa's rule format to describe OSes, CPU architectures, and file formats. The rule shown in Figure 2 uses `os` features to distinguish techniques used to get networking interface information on Windows and Linux. Note that the rule is explicit about which APIs are found on each OS, making it easy for both humans and machines to interpret the matching logic.

```
rule:
  meta:
    name: get networking interfaces
    namespace: host-interaction/network/interface
    author:
      - moritz.raabe@mandiant.com
      - joakim@intezer.com
    scope: function
    att&ck:
      - Discovery::System Network Configuration Discovery [T1016]
    examples:
      - B7841B9D5DC1F511A93CC7576672EC0C:0x1000EBF0
  features:
    - or:
      - and:
        - os: windows
        - api: iphlpapi.GetIfTable
        - api: iphlpapi.GetAdaptersInfo
      - and:
        - os: linux
        - api: getifaddrs
```

Figure 2: capa rule using the os feature to distinguish OS specific features

We've also added `arch` (such as `arch: i386` for 32-bit Intel code) and `format` (such as `format: elf` for ELF files) features to distinguish between CPU architectures and file formats. To learn more about these and capa's rule syntax see the rule format documentation on GitHub.

Unfortunately, rules with these new features are not backwards compatible with older versions of capa. Therefore, you should prefer to upgrade your capa installation to take advantage of our enhanced rules.

Substring Features

To make many rules easier to read, we've added a convenience feature named `substring` that acts like a literal string match with implied leading and trailing wildcards. This makes it easier to match file path components, such as `/.ssh/id_rsa`. Previously, users had to wrap a substring with forward slashes and escape special characters with backslashes, leading to nearly incomprehensible character sequences. Now, a substring feature clearly describes a literal string found as part of a longer string. Figure 3 shows how much easier it is to read a substring feature.

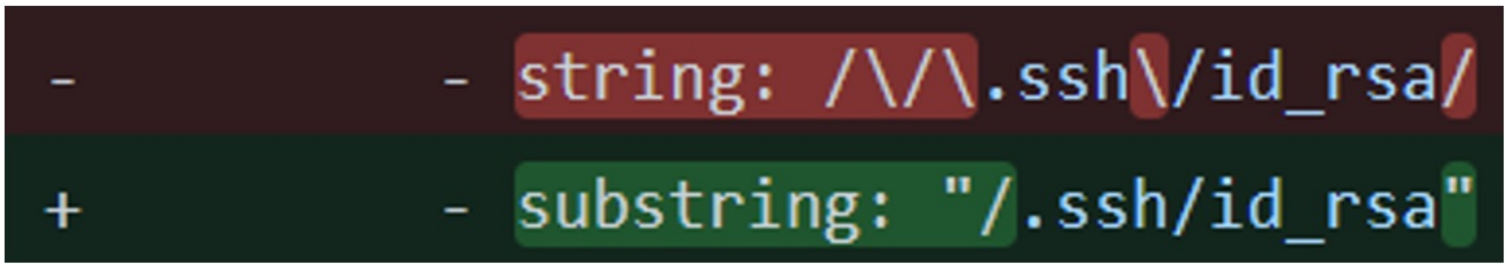

Figure 3: Old- and new-style ways of describing a substring

Figure 4 shows a capa rule using a substring feature to describe a persistence location on Linux.

```
rule:
  meta:
    name: persist via rc script
    namespace: persistence/service
    author: joakim@intezer.com
    scope: function
    att&ck:
      - Persistence::Boot or Logon Initialization Scripts::RC Scripts [T1037.004]
    examples:
      - 7351f8a40c5450557b24622417fc478d:0x407D11
  features:
    - and:
      - os: linux
      - match: host-interaction/file-system/write
      - or:
        - substring: "/etc/init.d/"
        - string: /\/etc\/rc[0-9].d\//
```

Figure 4: capa rule using the substring feature to identify persistence on Linux systems

Conclusion

The newest improvements add ELF file analysis support to capa and make its rules even more expressive. We thank the community and notably Intezer for their continued support. We love the collaboration and are excited for future opportunities. The v3 capa release also includes bug fixes, improvements to the IDAPython plugin capa explorer, and more than 50 new rules. See the capa changelog for all update details.

The new capa release is available on the release page and on PyPI. capa's code and rules are available on GitHub. If you have any questions or feedback, please open an issue or discussion in the respective repository.

NEXT >

Press Releases
Webinars
Events
Awards and Honors
Email Preferences

Technical Support
Incident?
Report Security Issue
Contact Support
Customer Portal
Communities
Documentation Portal

FireEye Blogs
Threat Research
FireEye Stories
Industry Perspectives

Threat Map
View the Latest Threats

Contact Us
+1 877-347-3393

Stay Connected

Privacy & Cookies Policy | Privacy Shield | Legal Documentation

Site Language
English