
Analysis Correlation Engine Documentation

Release 0.0.1

John Davison

Jul 26, 2019

Contents:

1	Major Features	3
1.1	Installation + Adding Data	4
1.1.1	Super fast How-To	4
1.1.2	Detailed Installation	4
1.1.3	Troubleshooting & Help	6
1.1.4	Getting Data into ACE	6
1.2	Analyst Orientation - Start Here	7
1.2.1	Quick Concept Touchpoint	7
1.2.2	GUI Overview	12
1.2.3	Working Alerts	12
1.3	ACE API Examples	20
1.3.1	Connect to a Server	22
1.3.2	Submitting data to ACE	22
1.3.3	Forcing Alert Creation	24
1.3.4	Downloading Cloudphish Results	25
1.3.5	Downloading an Alert	25
1.4	ACE API	25
1.4.1	Python Library	25
1.4.2	Common API	25
1.4.3	Alert API	25
1.5	Some Background	26
1.5.1	Driving Behavior	26
1.6	Additional Features	27
1.6.1	Events	27
1.6.2	Metrics	28
1.7	Administration Guide	28
1.7.1	Concepts	28
1.7.2	Turning on Engines	29
1.7.3	Enabling Modules	30
1.8	Development Guide	32
1.9	Developer README	32
1.9.1	SAQ = ACE	32
1.9.2	Everything was initially command line driven.	32
1.9.3	This was an internal project.	32
1.9.4	The database came later.	32
1.9.5	Unit testing.	33

1.9.6	Final words.	33
2	Indices and tables	35

Release v0.0.1.

ACE is a detection system and automation framework. At ACE's foundation are its engines for recursive analysis and its delivery of an intuitive presentation to the analyst. ACE's goal is to reduce the analyst's time-to-disposition to as close to zero as humanly possible.

While ACE is a powerful detection system, and does have built in detections, ACE does not ship with all of the yara signatures and intel detections that teams have built around it. However, ACE makes it easy to load your own yara signatures and atomic indicator detections.

Alerts are sent to ACE, and ACE handles the ordinary, manual, redundant, and repetitive tasks of collecting, combining, and relating data. ACE will then contextually and intuitively present all the right data to the analyst, allowing for a quick, high confidence determination to be made.

Got some new analysis that can be automated? Awesome! Add your automation, and let ACE keep working for you.

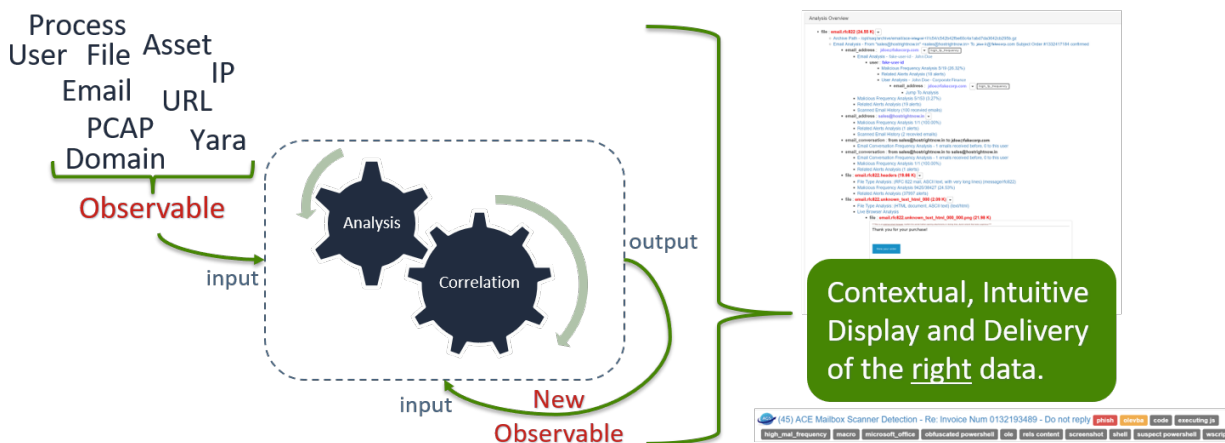


Fig. 1: Recursive Analysis; Presentation

For the most part, custom hunting tools send alerts to ACE using ACE's client library (API wrapper). ACE then gets to work by taking whatever detectable conditions it's given and spiraling out through its recursive analysis of observables, hitting as many detection points as possible across the attack surface.

Regardless of skill level, ACE greatly reduces the time it takes an analyst to make a high confidence determination, or as we call it, disposition. This reduction in time-to-disposition, coupled with the appropriate hunting and tuning mindset, means that security teams can greatly increase the attack surface they cover, all while utilizing the same amount of analyst time and practically eliminating alert fatigue. Optimization good, alert fatigue bad.

CHAPTER 1

Major Features

ACE is the implementation of a proven detection strategy, a framework for automating analysis, a central platform to launch and manage incident response activities, an email scanner, and much more.

- Email Scanning
- Recursive File Scanning
- URL Crawling and Content Caching
- Intuitive Alert Presentation
- Recursive Data Analysis & Correlation
- Central Analyst Interface
- Event/Incident management
- Intel Ingestion
- Modular Design for extending automation

An analyst



An analyst using ACE



1.1 Installation + Adding Data

1.1.1 Super fast How-To

1. Clean Ubuntu 18 install. Take a quick look at [these notes about Ubuntu 18](#).
2. Create username/group ace/ace.
3. Add ace to sudo.
4. Login as user ace.
5. `sudo mkdir /opt/ace && sudo chown ace:ace /opt/ace && cd /opt/ace`
6. `git clone https://github.com/IntegralDefense/ACE.git`.
7. `./installer/source_install`
8. `source load_environment`
9. `./ace add-user username email_address`
10. Goto <https://127.0.0.1/ace/> or whatever IP address you're using.

1.1.2 Detailed Installation

Install Ubuntu Server 18.04 LST

The size specifications for your server need to be based on your needs. At a minimum, the server should have 4 GB RAM and 20 GB storage drive. When installing the server, all of the default configurations are fine.

Getting Everything Ready

The ace User

```
$ sudo adduser ace
$ sudo adduser ace sudo
$ sudo su - ace
$ sudo chown ace:ace /opt
```

Cloning ACE

As the ace user you previously created, cd into /opt and git clone the IntegralDefense ACE master branch: <https://github.com/IntegralDefense/ACE.git>:

```
$ cd /opt
$ git clone https://github.com/IntegralDefense/ACE.git ace
```

Run the Installer

With *everything ready*, you can now run the ACE installer. Run the installer as the ace user. This will take a little while to complete.

```
$ cd /opt/ace
$ ./installer/source_install
```

Set Up Environment

Next, you will need to load the default environment variables ACE depends on. This load needs to be sourced from bash with the following command:

```
$ source load_environment
```

Create Users

Users are managed from the ACE command line with the following ace commands:

add-user	Add a new user to the system.
modify-user	Modifies an existing user on the system.
delete-user	Deletes an existing user from the system.

Create your first user so that you can log into the ACE GUI:

```
./ace add-user <username> <email_address>
```

Log into the GUI

You should now be able to browse to https://your_ip/ace/ and log into ACE with the user you previously created.

1.1.3 Troubleshooting & Help

There are a couple snags and gotchas that you can run into when installing ACE. This section will detail a few, but it's still a work in process. So, please send any issues or questions to ace-support@integraldefense.com. Please include as much detail as possible and we will get back to you as soon as we can. Thanks!

No Web GUI?

Make sure apache2 is running and the `/etc/apache2/sites-enabled/ace.conf` configuration is loaded. The `ace.conf` should be a symlink in `/etc/apache2/sites-available` that points to `/opt/ace/etc/saq_apache.conf`.

Alerts staying in 'NEW' status?

Make sure the ACE engine is running. You can do this by running the following:

```
cd /opt/ace && bin/start-correlation-engine
```

Start ACE

You should now have a working installation, but you need to start ACE's core (the correlation engine) this is accomplished with the `bin/start-correlation-engine` command. You can also use the `bin/start-ace` command, which start the correlation engine and attempt to start some other ACE collectors/services. You will get some errors if you don't have those other services configured (which you probably won't at this point). Those errors are nothing to be concerned about, however, if you do not want to see those errors you can explicitly start the correlation engine you need like so:

```
cd /opt/ace && bin/start-correlation-engine
```

1.1.4 Getting Data into ACE

A bare-bones ACE install is not going to be very effective by itself, much less without data. You can use the *Manual Analysis* section to submit observables to ACE. However, you're going to want to turn on some of the additional *Engines* and *Modules* that come with ACE by default. First, turning on the *Correlation Engine* is essential. Some other good engines to turn on first are the *CloudPhish* engine and the *Email Scanning* engine and if you've got yara signatures, definitely turn on the *Yara Scanner* module. See the *Administration Guide* for more details on the various engines, modules, and how to turn them on.

Manual Analysis

Via the Manual Analysis page, an analyst can submit an observable for ACE to analyze.

By default, the Insert Date is set to the current time, and the Description is set to 'Manual Correlation'. You can change the description to something meaningful. The Target Company will also be set to default, which should be fine for most ACE installations.

Alert Info

Insert Date 08-28-2018 15:53:57

Description Manual Correlation

Target Company default

Observables [Add](#)

Time (Optional)

Type file

Value [Choose File](#) No file chosen

[Submit](#)

Fig. 1: Observables can be submitted for analysis via the Manual Analysis page

Select the type of observable you wish to correlate and then provide the value. Click the Add button to correlate more than one observable type and/or value at a time.

Shortly after you've submitted your observable(s) for correlation, you will see your alert appear on the Manage Alerts page with the description you provided. The alert status will change to 'Complete' once ACE is finished performing its analysis. You must currently refresh the Manage Alerts page to see the alert status updates.

Using the API

ACE has an API that makes it simple to submit data to ACE for analysis and/or correlation. Check out the [ACE API Examples](#) and [ACE API](#) section for more information.

1.2 Analyst Orientation - Start Here

Keep this in mind when working ACE alerts: ACE is meant to enable the analyst to QUICKLY disposition false positive alerts and recognize true positives.

For convenience, here is a video recording that provides a tour of the ACE GUI and demonstrates how to work some ACE alerts. Many of the concepts in this orientation are covered.

1.2.1 Quick Concept Touchpoint

There are two core concepts an analyst must be familiar with when working ACE alerts: Observables and Dispositioning.

Observables

Observables are anything an analyst might "observe" or take note of during an investigation or when performing Alert Triage. For instance, an IP address is an observable, and a file name is a different type of observable. Some more

observable types are: URLs, domain names, usernames, file hashes, file names, file paths, email addresses, and Yara signatures.

ACE knows what kind of analysis to perform for a given observable type and how to correlate the value of an observable across all available data sources. In the process of correlating observables with other data sources, ACE will discover more observables to analyze and correlate.

When an ACE alert is created from an initial detection point, the alert's 'root' level observables are found in the output of that initial detection. ACE then gets to work on those root observables. An ACE alert's status is complete when ACE is finished with its recursive analysis, correlation, discovery, and relational combination of observables. The result is an ACE alert with intuitive context ready for the analyst's consumption.

The figure below is meant to give a visual representation ACE's recursive observable analysis and correlation.

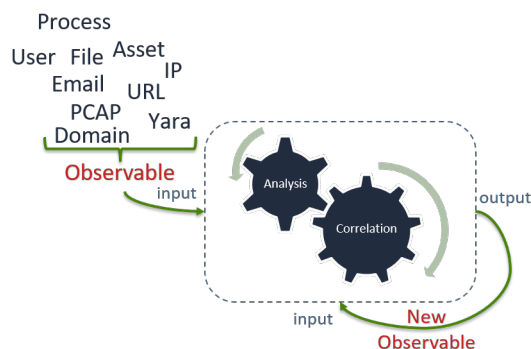


Fig. 2: Recursive Observable Analysis

ACE's recursive analysis of observables reduces and simplifies the analyst's workload by providing the analyst with as much available context as reasonably possible. A complete list of currently defined observable types can be viewed in the table below.

Currently defined ACE Observables:

Observable Type	Description
asset	An F_IPV4 identified to be a managed asset
email_address	Email address
email_conversation	A conversation between a source email address (MAIL FROM) and a destination email address (RCPT TO)
file	Path to an attached file
file_location	The location of file with format <code>hostname@full_path</code>
file_name	A file name (no directory path)
file_path	A file path
fqdn	Fully qualified domain name
hostname	Host or workstation name
indicator	CRITs indicator object ID
ipv4	IP address (version 4)
ipv4_conversation	Two F_IPV4 that were communicating formatted as <code>aaa.bbb.ccc.ddd_aaa.bbb.ccc.ddd</code>
md5	MD5 hash
message_id	Email Message-ID
process_guid	CarbonBlack global process identifier
sha1	SHA1 hash
sha256	SHA256 hash
snort_sig	Snort signature ID
url	A URL
user	An NT user ID identified to have used a given asset in the given period of time
yara_rule	Yara rule name

Alert Dispositioning

When investigating an alert, there is a categorization model for analysts to follow called dispositioning. No matter if an alert requires a response or not, analysts need to disposition them correctly. Sometimes, especially for true positive alerts that get escalated, more information may lead a change in an alert's disposition. The disposition model that ACE uses is based on Lockheed Martin's [Cyber Kill Chain](#)® model for identifying and describing the stages of an adversary's attack. The table below describes each of the different dispositions used by ACE.

Disposition	Description / Example
FALSE_POSITIVE	<p>Something matched a detection signature, but that something turned out to be nothing malicious.</p> <ul style="list-style-type: none"> A signature was designed to detect something specific, and this wasn't it. A signature was designed in a broad manner and, after analysis, what it detected turned out to be benign.
IGNORE	<p>This alert should have never fired. A match was made on something a detection was looking for but it was expected or an error.</p> <ul style="list-style-type: none"> Security information was being transferred An error occurred in the detection software generating invalid alerts Someone on the security team was testing something or working on something <p>It is important to make the distinction between FALSE_POSITIVE and IGNORE dispositions, as alerts marked FALSE_POSITIVE are used to tune detection signatures, while alerts marked as IGNORE are not. IGNORE alerts are deleted by cleanup routines.</p>
UNKNOWN	Not enough information is available to make a good decision because of a lack of visibility.
REVIEWED	<p>This is a special disposition to be used for alerts that were manually generated for analysis or serve an informational purpose. For example, if someone uploaded a malware sample from a third party to ACE, you would set the disposition to REVIEWED after reviewing the analysis results. Alerts set to REVIEWED do not count for metrics and are not deleted by cleanup routines.</p>
GRAYWARE	<p>Software that is not inherently malicious but exhibits potentially unwanted or obtrusive behavior.</p> <ul style="list-style-type: none"> Adware Spyware <p>If desired, this disposition can be used to categorize spam emails.</p>
POLICY_VIOLATION	<p>In the course of an investigation, general risky user behavior or behavior against an official policy or standard is discovered.</p> <ul style="list-style-type: none"> Installing unsupported software Connecting a USB drive with pirated software Browsing to pornographic sites
RECONNAISSANCE	<p>Catching the adversary planning, gathering intel, or researching what attacks may work against you.</p> <ul style="list-style-type: none"> Vulnerability and port scanning Attempts to establish trust with a user
WEAPONIZATION	<p>The detection of an attempt to build a cyber attack weapon.</p> <ul style="list-style-type: none"> Detecting an adversary building a malicious document using VT threat hunting
DELIVERY	<p>An attack was attempted, and the attack's destination was reached. Even with no indication the attack worked!</p> <ul style="list-style-type: none"> A user browsed to an exploit kit A phish was delivered to the email inbox AV detected and remediated malware after the

1.2. Analyst Orientation - Start Here

1.2.2 GUI Overview

Analysts interact with ACE through its graphical interface and specifically use the Manage Alerts page. After you're logged into ACE (Assuming you already have an account), you'll see a navigation bar that looks like the following image. A simple breakdown of each page on that navigation bar is provided below.

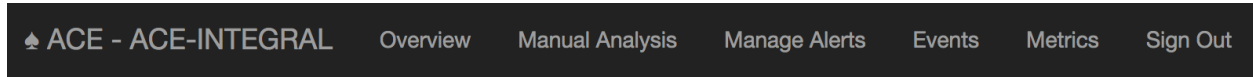


Fig. 3: ACE's Navigation Bar

Page	Function
Overview	General ACE information, performance, statistics, etc.
Manual Analysis	Where analysts can manually upload or submit observables for ACE to analyze
Manage Alerts	The alert queue - where the magic happens
Events	Where events are managed
Metrics	For creating and tracking metrics from the data ACE generates

1.2.3 Working Alerts

This section covers the basics for working and managing ACE alerts. If you're comfortable, skip ahead to the [Examples](#) section to find a walkthrough of a few ACE alerts being worked.

The Manage Alerts Page

ACE alerts will queue up on the Manage Alerts page. By default, only alerts that are open (not *dispositioned*) and not owned by another analyst are displayed. When working an alert, analysts should **take ownership** of it to prevent other analysts from starting to work on the same alert. This prevents re-work and saves analyst time. You can take ownership of one or more alerts on the Manage Alerts page by selecting alert checkboxes and clicking the 'Take Ownership' button. You can also take ownership when viewing an individual alert. Below is an example of the Manage Alerts page with 32 open and unowned alerts.

Viewing Observable Summary

On the Manage Alerts page, each alert can be expanded via its dropdown button. Once expanded, all the observables in the alert can be viewed. The observables are grouped and listed by their observable type. The numbers in parentheses show a count of how many times ACE has seen that observable. Each observable is clickable, and when clicked, ACE will add that observable to the current alert filter. You don't need to worry about alert filtering to work alerts, however, the [Filtering and Grouping](#) section covers Alert filtering.


Expand/Collapse Observables

```

- email_address
  - fakeuser@fakecompany.com (21)
  - tfry@kennyross.com (2)
- email_conversation
  - tfry@kennyross.com|fakeuser@fakecompany.com (1)
- file
  - 308591a9db1d3b8739e53feaf3dd5ba069f7191125cf3bb7e2c849bad2182e98.vxstream/dropped/
  - 1LSZPI0TG6C82HTABETK.temp (1)
  - 308591a9db1d3b8739e53feaf3dd5ba069f7191125cf3bb7e2c849bad2182e98.vxstream/dropped/
  - Kenny_Ross_Inquiry.LNK (1)

```

(continues on next page)


2018-08-21 08:32:55 (45) ACE Mailbox Scanner Detection - Re: Invoice Num 0132193489 - Do not reply
phish
olevba
code
executing js

high_mal_frequency
macro
microsoft_office
obfuscated powershell
ole
rels content
screenshot
shell
suspect powershell
wscript

- email_address
 - tfr@kennyross.com (2)
 - @.com (9)
- email_conversation
 - tfr@kennyross.com|@.com (1)
- file
 - Kenny_Ross_Inquiry.doc (9)
 - Kenny_Ross_Inquiry.doc.officeparser/iYzcZYMdfv.bas (2)
 - Kenny_Ross_Inquiry.doc.officeparser/oUDOGruwp.bas (2)
 - Kenny_Ross_Inquiry.doc.officeparser/stream_10_0.dat (2)
 - Kenny_Ross_Inquiry.doc.officeparser/stream_11_0.dat (2)
 - Kenny_Ross_Inquiry.doc.officeparser/stream_12_0.dat (2)
 - Kenny_Ross_Inquiry.doc.officeparser/stream_13_0.dat (2)
 - Kenny_Ross_Inquiry.doc.officeparser/stream_14_0.dat (2)
 - Kenny_Ross_Inquiry.doc.officeparser/stream_15_0.dat (2)
 - Kenny_Ross_Inquiry.doc.officeparser/stream_16_0.dat (2)
 - Kenny_Ross_Inquiry.doc.officeparser/stream_17_0.dat (2)

1.2. Analyst Orientation - Start Here

(continued from previous page)

```

- 308591a9db1d3b8739e53feaf3dd5ba069f7191125cf3bb7e2c849bad2182e98.vxstream/dropped/
↪index.dat (1)
- 308591a9db1d3b8739e53feaf3dd5ba069f7191125cf3bb7e2c849bad2182e98.vxstream/dropped/
↪urlref_httpvezopilan.comtstindex.php1_soho7.tkn_.Split (1)
- Kenny_Ross_Inquiry.doc (9)
- Kenny_Ross_Inquiry.doc.officeparser/iYzcZYMdfv.bas (2)
- Kenny_Ross_Inquiry.doc.officeparser/oUDOGruwp.bas (2)
- Kenny_Ross_Inquiry.doc.officeparser/stream_10_0.dat (2)
- Kenny_Ross_Inquiry.doc.officeparser/stream_11_0.dat (2)
- Kenny_Ross_Inquiry.doc.officeparser/stream_12_0.dat (2)
- Kenny_Ross_Inquiry.doc.officeparser/stream_13_0.dat (2)
- Kenny_Ross_Inquiry.doc.officeparser/stream_14_0.dat (2)
- Kenny_Ross_Inquiry.doc.officeparser/stream_15_0.dat (2)
- Kenny_Ross_Inquiry.doc.officeparser/stream_16_0.dat (2)
- Kenny_Ross_Inquiry.doc.officeparser/stream_17_0.dat (2)
- Kenny_Ross_Inquiry.doc.officeparser/stream_18_0.dat (2)
- Kenny_Ross_Inquiry.doc.officeparser/stream_19_0.dat (2)
- Kenny_Ross_Inquiry.doc.officeparser/stream_1_0.dat (2)
- Kenny_Ross_Inquiry.doc.officeparser/stream_2_0.dat (2)
- Kenny_Ross_Inquiry.doc.officeparser/stream_2_0.dat.extracted/WXRIK/WXRIK/WXRIK1.
↪lrA (2)
- Kenny_Ross_Inquiry.doc.officeparser/stream_2_0.dat.extracted/WXRIK/WXRIK/
↪WXRIKManager.lrA (2)
- Kenny_Ross_Inquiry.doc.officeparser/stream_2_0.dat.extracted/WXRIK/WXRIK/_pPOR/
↪WXRIKManager.lrA.pPOR (2)
- Kenny_Ross_Inquiry.doc.officeparser/stream_2_0.dat.extracted/[Content_Types].lrA_
↪(2)
- Kenny_Ross_Inquiry.doc.officeparser/stream_2_0.dat.extracted/_pPOR/.pPOR (2)
- Kenny_Ross_Inquiry.doc.officeparser/stream_3_0.dat (2)
- Kenny_Ross_Inquiry.doc.officeparser/stream_4_0.dat (2)
- Kenny_Ross_Inquiry.doc.officeparser/stream_5_0.dat (2)
- Kenny_Ross_Inquiry.doc.officeparser/stream_8_0.dat (2)
- Kenny_Ross_Inquiry.doc.officeparser/stream_9_0.dat (2)
- Kenny_Ross_Inquiry.doc.olevba/macro_0.bas (2)
- Kenny_Ross_Inquiry.doc.olevba/macro_1.bas (2)
- Kenny_Ross_Inquiry.doc.pcode.bas (2)
- email.rfc822 (37952)
- email.rfc822.headers (37949)
- email.rfc822.unknown_text_html_000 (3229)
- email.rfc822.unknown_text_html_000_000.png (2482)
- email.rfc822.unknown_text_plain_000 (37354)
- filename.PNG (11)
- indicator
- 55c36786bcb87f2d54cf15da (369)
- 57ffd02cbcb87fbb1464b1ce (88)
- 58c9708aad951d7387c65be2 (274)
- 58e3e8dfad951d49aabb1622 (384)
- 58ee209dad951d09a1ee3860 (92)
- 58ee221dad951d09a0b13e99 (92)
- 5937f5d4ad951d4fe8787c63 (672)
- 599db056ad951d5cb2c4768b (302)
- 599dd8abad951d5cb3204569 (155)
- 59a7fcc7ad951d522eeef8ed (380)
- ipv4
- 104.118.208.249 (24)
- md5
- 2307a1a403c6326509d4d9546e5f32ab (2)

```

(continues on next page)

(continued from previous page)

```

- 267b1bd0ae8194781c373f93c9df02fa (2)
- 39ee938f6fa351f94a2cbf8835bb454f (2)
- 5c4c76cbb739c04fb3838aff5b2c25bb (2)
- 65811d8f7c6a1b94eab03ba1072a3a7e (2)
- b3b8bf4ed2c5cb26883661911487d642 (2)
- d8a7ea6ba4ab9541e628452e2ad6014a (2)
- message_id
- <8de41f6eb57ac01b2a90d3466890b0a1@127.0.0.1> (1)
- sha1
- 03484a568871d494ad144ac9597e9717a2ae5601 (2)
- 2e3b95bb9b0beb5db3487646d772363004505df6 (2)
- 33b9d3de33adc5bd5954c1e9f9e48f10eabe7c49 (2)
- 62837876eb5ec321e6d8dbd6babd0d5789230b60 (2)
- b3024c6f598b1745ca352ac3a24cc3603b814cad (2)
- cfe4f07fbf042b4f7dce44f9e6e3f449e02c123a (2)
- fa47ebc1026bbe8952f129480f38a011f9faf47d (2)
- sha256
- 308591a9db1d3b8739e53feaf3dd5ba069f7191125cf3bb7e2c849bad2182e98 (2)
- 50aef060b9192d5230be21df821acb4495f7dc90416b2edfd68ebdbde40562be (2)
- 62be2fe5e5ad79f62671ba4b846a63352d324bb693ee7c0f663f488e25f05fe0 (2)
- 8159227eb654ef2f60eb4c575f4a218bb76919ea15fdd625c2d01d151e4973f3 (2)
- 9c7e06164ec59e76d6f3e01fa0129607be1d98af270a09fd0f126ee8e16da306 (2)
- ae67f33b6ff45aecf91ff6cac71b290c27f791ccbe4829be44bd64468cbe3f5d (2)
- ca797ec10341aebaed1130c4dbf9a5b036945f17dd94d71d46f2f81d9937504f (2)
- url
- http://schemas.openxmlformats.org/drawingml/2006/main (3796)
- user
- fake_user_id (17)
- yara_rule
- CRITS_EmailContent (4478)
- CRITS_StringOffice (1685)
- CRITS_StringVBS (6592)
- CRITS_StringWindowsShell (1770)
- macro_code_snippet (1013)
- macro_overused_legit_functions (82)

```

Above, you can click to expand a text based example of an alerts observable structure when expanded on the Manage Alerts page.

Filtering and Grouping

On the *Manage Alerts* page, alerts are filtered by default to show open alerts that are not currently owned by any other analysts. The current filter state is always displayed at the top of the page, in a human readable format. You can select 'Edit Filters' to modify the alert filter and display alerts based on several different conditions. For example, you can change the filters to see alerts dispositioned as DELIVERY over the past seven days by a specific analyst.

Alerts can also be filtered by observables. Conveniently, when viewing an alert's *Observable Summary* on the Manage Alerts page, you can click any of those observables to add it to the currently defined alert filter. So, with the default filter applied, if you clicked on an MD5 observable with value *10EFE4369EA344308416FB59051D5947* then the page would refresh and you'd see that the new filter became:

```

filter: open alerts AND not owned by others AND with observable type md5 value b
→ '10EFE4369EA344308416FB59051D5947'`

```

The Alert Page

Once an alert is opened, the full analysis results will be displayed. It's usually a good idea to go ahead and [view](#) all of the alert's analysis.

Views

There are two different modes in which you can view ACE alerts: 'Critical' and 'All'. By default, ACE alerts will be displayed in critical mode. Critical mode will only display 'root' level alert observable analysis. This is helpful for alerts with a lot of observables, although it's generally helpful to view all alert analysis. At the top right of every alert you will see a button to "View All Analysis" or "View Critical Analysis". Whichever mode you have enabled will be persistent across your ACE session.

Be mindful of these different views, as it's possible for an analyst to miss helpful information if viewing an alert in critical mode compared to all mode.

Analysis Overview

Each standard ACE alert will have analysis overview section where the analysis results for every [Observable](#) will be found. The observables displayed at the 'root' level are the ones that were directly discovered in the data provided to ACE at the time of the alert's creation. Underneath each observable you will find the analysis results for that respective observable. You may also find new observables that were added to the alert from the recursive analysis of other observables. This observable nesting on the alert page provides a visual representation of how alert observables are related. The figure below shows the analysis overview section of an ACE Mailbox (email) alert. You can see that a user observable of value 'fake-user-id' was discovered from the analysis results of the email_address Observable.

Alert Tags

ACE has a tagging system by which observables and analysis are tagged for the purpose of providing additional context. If you review the previous figure of [Manage Alerts page](#), you will notice tags such as phish, new_sender, and frequent_conversation associated to various alerts.

All observable tags get associated with their respective alert and show up on the alert management page. Any observable can be tagged and can have any number of tags. For instance, an email conversation between two addresses that ACE has seen a lot will be tagged as 'frequent_conversation'. Tags can also be added directly to alerts from the Manage Alerts page. This can be helpful for [Filtering and Grouping](#) alerts if an analyst needs a way to group alerts that don't otherwise have a commonly shared tag or observable.

Examples

The following are examples of a snarky analyst working ACE alerts. Think about the first intuition you get from what you see in these alerts.

Check out this Email Alert

We just got this alert in the queue. Huh, looks like this email might be related to a potentially malicious zip file.

Let's open the alert and look at the Analysis Overview section to see the results ACE brought us. In the case of email alerts like this one, the 'email.rfc882' file is what ACE was given when told to create this alert.

Under that email.rfc882 file observable you will see the output of the Email Analysis module, and underneath Email Analysis you will see where ACE discovered more observables, such as the email addresses.

Analysis Overview

- file : **email.rfc822 (24.55 K)**
 - Archive Path - /opt/saq/archive/email/ace-.../c54/c542b42fbe60c4a1abd7da3642cb295b.gz
 - Email Analysis - From "sales@hostrightnow.in" <sales@hostrightnow.in> To ...@...com Subject Order #1332417184 confirmed
 - email_address : **jdoo@fakecorp.com** high_fp_frequency
 - Email Analysis - fake-user-id - John Doe
 - user : **fake-user-id**
 - Malicious Frequency Analysis 5/19 (26.32%)
 - Related Alerts Analysis (18 alerts)
 - User Analysis - John Doe - Corporate Finance
 - email_address : **jdoo@fakecorp.com** high_fp_frequency
 - Jump To Analysis
 - Malicious Frequency Analysis 5/153 (3.27%)
 - Related Alerts Analysis (19 alerts)
 - Scanned Email History (100 received emails)
 - email_address : **sales@hostrightnow.in**
 - Malicious Frequency Analysis 1/1 (100.00%)
 - Related Alerts Analysis (1 alerts)
 - Scanned Email History (2 received emails)
 - email_conversation : from sales@hostrightnow.in to jdoo@fakecorp.com
 - Email Conversation Frequency Analysis - 1 emails received before, 0 to this user
 - email_conversation : from sales@hostrightnow.in to sales@hostrightnow.in
 - Email Conversation Frequency Analysis - 1 emails received before, 0 to this user
 - Malicious Frequency Analysis 1/1 (100.00%)
 - Related Alerts Analysis (1 alerts)
 - file : **email.rfc822.headers (19.66 K)**
 - File Type Analysis: (RFC 822 mail, ASCII text, with very long lines) (message/rfc822)
 - Malicious Frequency Analysis 9425/38427 (24.53%)
 - Related Alerts Analysis (37997 alerts)
 - file : **email.rfc822.unknown_text_html_000 (2.09 K)**
 - File Type Analysis: (HTML document, ASCII text) (text/html)
 - Live Browser Analysis
 - file : **email.rfc822.unknown_text_html_000_000.png (21.98 K)**

This is an external email message. Confirm the sender before opening attachments or clicking links. Avoid content that looks suspicious.

Thank you for your purchase!

[View your order](#)

Fig. 6: The Analysis Overview section of an email alert

(5) ACE Mailbox Scanner Detection - Order #1332417184 confirmed **malicious** **redirection_target** **screenshot** **single_file_zip** **zip** **exe_in_zip**

Analysis Overview

- file : **email.rfc822 (24.55 K)**
 - Archive Path - /opt/saq/archive/email/ace-[redacted]/c54/c542b42fbe60c4a1abd7da3642cb295b.gz
 - Email Analysis - From "sales@hostrightnow.in" <sales@hostrightnow.in> To [redacted]@[redacted].com Subject Order #1332417184 confirmed
 - email_address : **jdoe@fakecorp.com**
 - Email Analysis - fake-user-id - John Doe
 - user : **fake-user-id**
 - Malicious Frequency Analysis 5/19 (26.32%)
 - Related Alerts Analysis (18 alerts)
 - User Analysis - John Doe - Corporate Finance
 - email_address : **jdoe@fakecorp.com**
 - Jump To Analysis
 - Malicious Frequency Analysis 5/153 (3.27%)
 - Related Alerts Analysis (19 alerts)
 - Scanned Email History (100 received emails)
 - email_address : **sales@hostrightnow.in**
 - Malicious Frequency Analysis 1/1 (100.00%)
 - Related Alerts Analysis (1 alerts)
 - Scanned Email History (2 received emails)
 - email_conversation : from sales@hostrightnow.in to jdoe@fakecorp.com
 - Email Conversation Frequency Analysis - 1 emails received before, 0 to this user
 - email_conversation : from sales@hostrightnow.in to sales@hostrightnow.in
 - Email Conversation Frequency Analysis - 1 emails received before, 0 to this user
 - Malicious Frequency Analysis 1/1 (100.00%)
 - Related Alerts Analysis (1 alerts)
 - file : **email.rfc822.headers (19.66 K)**
 - File Type Analysis: (RFC 822 mail, ASCII text, with very long lines) (message/rfc822)
 - Malicious Frequency Analysis 9425/38427 (24.53%)
 - Related Alerts Analysis (37997 alerts)
 - file : **email.rfc822.unknown_text_html_000 (2.09 K)**
 - File Type Analysis: (HTML document, ASCII text) (text/html)
 - Live Browser Analysis
 - file : **email.rfc822.unknown_text_html_000_000.png (21.98 K)**

This is an external email message. Confirm the sender before opening attachments or clicking links. Avoid content that looks suspicious.


Thank you for your purchase!

[View your order](#)

 (1) ACE Mailbox Scanner Detection - New Application For Retail Real Estate Development Manager microsoft_office new_sender zip

When we open the alert, we see the alert header at the top. Hmm, this email alert only has one detection. Either this is really good phish and something we barely catch, or it's a false positive.

ACE Mailbox Scanner Detection - New Application For Retail Real Estate Development Manager

Company	 Default (1)
Alert Time	2018-09-22 09:52:05
Source	ACE - Mailbox Scanner
Instance	open-ace
Alert Type	mailbox
Storage	data/localhost.localdomain/e5f/e5f0f475-1498-40f0-ab69-f585d034005e
Priority	0
Status	Completed
Detections	1 detections

Let's scroll down and find that single detection. Oh, I just noticed that we're only viewing this alert's critical analysis.

We could click on the "View All Analysis" button if we wanted to view all of its analysis results. However, for this alert, the critical view makes it easy to find the single detection. Detections are marked by a little red flame icon. Here we see that the flame is highlighting a yara rule that detected something in the analysis of the "Glenn Resume.docx" file. Speaking of that file, we were right about assuming it was an open xml office document.

Look at this, ACE tagged the *rels_remote_references* yara rule with *high_fp_frequency*. That tells us that this specific yara rule has a high frequency of showing up in false positive alerts. Below the rule, we see that the "Malicious Frequency Analysis" module found the *rels_remote_references* yara rule only appeared in four true-positive alerts out of two hundred and ten! I don't know about you, but my gut is telling me this email alert is a false positive. Let's make sure though and click to view the "Yara Scan Results".

Above we can see what the yara rule detected in this docx file. And what do we see? A target reference to a file, and when looking closer we see that the file being referenced was named "Resume Template.dotm". I bet this dotm file is a leftover artifact from Glenn using a resume template when creating this "Glenn Resume.docx" file. I'm already clicking the "Disposition" button and marking this alert FALSE_POSITIVE.

Now that we've reviewed this email alert, I want to harp on how QUICKLY we should be able to disposition it.

If you're curious, the *rels_remote_references* yara rule was created to detect references to URLs or files in an open xml document's template. Such references can and have been malicious. An example would be a Microsoft Word document that references a URL and causes word to display an authentication dialog to the end-user for the purpose of harvesting the user's credentials. This repository contains a GO script that makes it easy to do that very thing: <https://github.com/ryhanson/phishery>

1.3 ACE API Examples

Let's go through a few examples using the ACE API. We will specifically use the `ace_api` python library.

Critical Analysis Overview *

- file : **email.rfc822 (46.63 K)**
 - Email Analysis - From ace@open-ace.RAC2V1A To ace@open-ace Subject New Application For Retail Real Estate Development Manager
 - email_address : **ace@open-ace**
 - Malicious Frequency Analysis 0/7 (0.00%)
 - Related Alerts Analysis (17 alerts)
 - email_address : **ace@open-ace.rac2v1a**
 - Malicious Frequency Analysis 0/1 (0.00%)
 - Related Alerts Analysis (4 alerts)
 - email_conversation : from ace@open-ace.RAC2V1A to ace@open-ace new_sender
 - Email Conversation Frequency Analysis - first time received
 - Malicious Frequency Analysis 0/1 (0.00%)
 - Related Alerts Analysis (4 alerts)
 - file : **Glenn Resume.docx (29.9 K)** microsoft_office zip
 - Archive Analysis (28 files available 28 extracted)
 - file : **Glenn Resume.docx.extracted/word/_rels/settings.xml.rels (395 bytes)**
 - File Type Analysis: (XML 1.0 document, ASCII text, with very long lines, with CRLF line terminators) (text/xml)
 - Malicious Frequency Analysis 0/1 (0.00%)
 - Related Alerts Analysis (1 alerts)
 - Yara Scan Results: 1 results
 - yara_rule : **rels_remote_references** high_fp_frequency
 - Malicious Frequency Analysis 4/210 (1.90%)
 - Related Alerts Analysis (220 alerts)
 - File Hash Analysis 3420f670c36f09c0830db4777bc9b838494a42b754b8ab91b8ab0a134aa71d07
 - File Type Analysis: (Microsoft Word 2007+) (application/vnd.openxmlformats-officedocument.wordprocessingml.document)
 - Malicious Frequency Analysis 0/1 (0.00%)
 - Related Alerts Analysis (1 alerts)
 - File Type Analysis: (RFC 822 mail, ASCII text) (message/rfc822)
 - Malicious Frequency Analysis 0/7 (0.00%)
 - Related Alerts Analysis (17 alerts)

rels_remote_references

target /opt/ace/work/email_scanner/e5f/e5f0f475-1498-40f0-ab69-f585d034005e/Glenn Resume.docx.extracted/word/_rels/settings.xml.rels

meta {'file_ext': 'rels'}

String Matches

position 257 string \$\$b1 value Target="file://

```

00000000 66 6f 72 6d 61 74 73 2e 6f 72 67 2f 6f 66 66 69 |formats.org/offi|
00000010 63 65 44 6f 63 75 6d 65 6e 74 2f 32 30 30 36 2f |ceDocument/2006/|
00000020 72 65 6c 61 74 69 6f 6e 73 68 69 70 73 2f 61 74 |relationships/at|
00000030 74 61 63 68 65 64 54 65 6d 70 6c 61 74 65 22 20 |tachedTemplate" |
00000040 54 61 72 67 65 74 3d 22 66 69 6c 65 3a 2f 2f 2f |Target="file:///|
00000050 2f 43 3a 5c 55 73 65 72 73 5c 72 69 63 6f 5c 41 |/C:\Users
ico\A|
00000060 70 70 44 61 74 61 5c 52 6f 61 6d 69 6e 67 5c 4d |ppData\Roaming\M|
00000070 69 63 72 6f 73 6f 66 74 5c 54 65 6d 70 6c 61 74 |icrosoft\Templat|
00000080 65 73 5c 42 61 73 69 63 25 32 30 52 65 73 75 |es\Basic%20Resu|
0000008f

```

position 341 string \$\$c2 value Template.dotm

```

00000000 55 73 65 72 73 5c 72 69 63 6f 5c 41 70 70 44 61 |Users
ico\AppData|
00000010 74 61 5c 52 6f 61 6d 69 6e 67 5c 4d 69 63 72 6f |ta\Roaming\Micro|
00000020 73 6f 66 74 5c 54 65 6d 70 6c 61 74 65 73 5c 42 |soft\Templates\B|
00000030 61 73 69 63 25 32 30 52 65 73 75 6d 65 25 32 30 |asic%20Resume%20|
00000040 54 65 6d 70 6c 61 74 65 2e 64 6f 74 6d 22 20 54 |Template.dotm" T|
00000050 61 72 67 65 74 4d 6f 64 65 3d 22 45 78 74 65 72 |targetMode="Exter|

```

1.3.1 Connect to a Server

By default, the `ace_api` library will attempt to connect to `localhost`. Use the `ace_api.set_default_remote_host()` function to have the library connect to a different server. The OS's certificate store is used to validate the server. See `ace_api.set_default_ssl_ca_path()` to change this behavior.

```
>>> import ace_api

>>> server = 'ace.integraldefense.com'

>>> ace_api.set_default_remote_host(server)

>>> ace_api.ping()
{'result': 'pong'}
```

You can over-ride this default in the `ace_api.Analysis()` class with the `ace_api.Analysis.set_remote_host()` method and you can also manually specify a remote host with any submit.

```
>>> analysis = ace_api.Analysis('this is the analysis description')

>>> analysis.remote_host
'ace.integraldefense.com'

>>> analysis.set_remote_host('something.else.com').remote_host
'something.else.com'

>>> ace_api.default_remote_host
'ace.integraldefense.com'
```

If your ACE instance is listening on a port other than 443, specify it like so:

```
>>> ace_api.set_default_remote_host('ace.integraldefense.com:24443')

>>> ace_api.default_remote_host
'ace.integraldefense.com:24443'
```

1.3.2 Submitting data to ACE

You should submit data to ace by first creating an *Analysis* object and loading it with the data you want to submit for analysis and/or correlation. The below examples show how to perform some common submissions.

Submit a File

Say we have a suspect file in our current working director named “Business.doc” that we want to submit to ACE. First, we create an analysis object and then we pass the path to the file to the `ace_api.Analysis.add_file()` method. We will also include some tags and check the status (`ace_api.Analysis.status()`) of the analysis as ACE works on the submission.

```
>>> path_to_file = 'Business.doc'

>>> analysis.add_file(path_to_file)
<ace_api.Analysis object at 0x7f23d57e74e0>

>>> analysis.add_tag('Business.doc').add_tag('suspicious doc')
```

(continues on next page)

(continued from previous page)

```

<ace_api.Analysis object at 0x7f23d57e74e0>

>>> analysis.submit()
<ace_api.Analysis object at 0x7f23d57e74e0>

>>> analysis.status
'NEW'

>>> analysis.status
'ANALYZING'

>>> analysis.status
'COMPLETE (Alerted with 8 detections)'

>>> result_url = 'https://{}/ace/analysis?direct={}'.format(analysis.remote_host, _
↳analysis.uuid)

>>> print("\nThe results of this submission can be viewed here: {}".format(result_
↳url))

```

The results of this submission can be viewed here: <https://ace.integraldefense.com/ace/analysis?direct=137842ac-9d53-4a25-8066-ad2a1f6cfa17>

Submit a URL

Two examples of submitting a URL to ACE follow. The first example shows how to submit a URL by adding the URL as an observable to an *Analysis* object. This also allows us to demonstrate the use of directives. The second example shows how simple it is to submit a URL for analysis directly to Cloudphish.

As an observable

You can submit as many *observables* as you desire in a submission to ACE, but they won't necessarily get passed to every analysis module that can work on them by default. This is the case for URL observables, which by themselves, require the crawl directive to tell ACE you want to download the content from the URL for further analysis.

Submitting a request for a suspicious URL to be analyzed, note the use of the crawl directive and how to get a list of the valid directives.

```

>>> suspicious_url = 'http://davidcizek.cz/Invoice/ifKgg-jrzA_PvC-a7'

>>> analysis = ace_api.Analysis('Suspicious URL')

>>> analysis.add_tag('suspicious_url')
<ace_api.Analysis object at 0x7f23d57e7588>

>>> for d in ace_api.get_valid_directives()['result']:
...     if d['name'] == 'crawl':
...         print(d['description'])
...
crawl the URL

>>> analysis.add_url(suspicious_url, directives=['crawl']).submit()
<ace_api.Analysis object at 0x7f23d57e7588>

```

(continues on next page)

(continued from previous page)

```
>>> analysis.status
'COMPLETE (Alerted with 9 detections)'

>>> result_url = 'https://{}/ace/analysis?direct={}'.format(analysis.remote_host,
↳analysis.uuid)

>>> print("\nThe results of this submission can be viewed here: {}".format(result_
↳url))
```

The results of this submission can be viewed here: <https://ace.integraldefense.com/ace/analysis?direct=de66b2d3-f273-4bdd-a05b-771ecf5c8a76>

Using Cloudphish

If you just want ACE to analyze a single URL, it's best to submit directly to Cloudphish. In this example, a URL is submitted to cloudphish that cloudphish has never seen before and a 'NEW' status is returned. After cloudphish has finished analyzing the URL, the status changes to 'ANALYZED' and the `analysis_result` tells us at least one detection was found (as we alerted).

```
>>> another_url = 'http://medicci.ru/myATT/tu8794_QcbkoEsv_Xw20pYh7ij'

>>> cp_result = ace_api.cloudphish_submit(another_url)

>>> cp_result['status']
'NEW'

>>> # Query again, a moment later:
...
>>> cp_result = ace_api.cloudphish_submit(another_url)

>>> cp_result['status']
'ANALYZED'

>>> cp_result['analysis_result']
'ALERT'

>>> result_url = 'https://{}/ace/analysis?direct={}'.format(ace_api.default_remote_
↳host, cp_result['uuid'])

>>> print("\nThe results of this submission can be viewed here: {}".format(result_
↳url))
```

The results of this submission can be viewed here: <https://ace.integraldefense.com/ace/analysis?direct=732ec396-ce20-463f-82b0-6b043b07f941>

1.3.3 Forcing Alert Creation

By default, ACE alerts are only created if an detection is made in the initially submitted analysis. You can force alert creation by changing the default analysis mode from *analysis* to *correlation*. This is accomplished like so:

```
>>> analysis = ace_api.Analysis('This is an analysis with no detections', analysis_
↳mode='correlation')
```

(continues on next page)

(continued from previous page)

```
>>> analysis.submit()
<ace_api.Analysis object at 0x7fbe81af66a0>

>>> analysis.status
'COMPLETE (Alerted with 0 detections)'
```

1.3.4 Downloading Cloudphish Results

Cloudphish keeps a cache of the URL content it downloads. In this example we will download the results of the URL submitted in the previous example, which in this case is a malicious word document.

```
>>> ace_api.cloudphish_download(another_url, output_path='cp_result.raw')
True
>>> os.path.exists('cp_result.raw')
True
```

1.3.5 Downloading an Alert

You can use the `ace_api.download()` function to download an entire Alert. Below, we download an entire Alert and have it written to a directory named by the Alert's UUID.:

```
>>> uuid = cp_result['uuid']

>>> >>> uuid
'732ec396-ce20-463f-82b0-6b043b07f941'

>>> ace_api.download(uuid, target_dir=uuid)
```

Now, there is a new directory named '732ec396-ce20-463f-82b0-6b043b07f941' in our current working directory that contains all of the files and data from the alert with uuid 732ec396-ce20-463f-82b0-6b043b07f941. Use the `ace_api.load_analysis()` function to load an alert into a new *Analysis* object.

1.4 ACE API

1.4.1 Python Library

A python library exists for interacting with the ACE API. You can install it with pip: `pip3 install ace_api`.

1.4.2 Common API

1.4.3 Alert API

submit

Submits a new alert to ACE. These go directly into the correlation engine for analysis and show up to analysts as alerts.

Parameters: alert - JSON dict with the following schema

```
{
  'tool': tool_name,
  'tool_instance': tool_instance_name,
  'type': alert_type,
  'description': alert_description,
  'event_time': time of the alert/event (in %Y-%m-%dT%H:%M:%S.%f%z format),
  'details': free-form JSON dict of anything you want to include,
  'observables': (see below),
  'tags': a list of tags to add to the alert,
}
```

The observables field is a list of zero or more dicts with the following format

```
{
  'type': The type of the observable,
  'value': The value of the observable,
  'time': The optional time of the observable (can be null),
  'tags': Optional list of tags to add to the observable,
  'directives': Optional list of directives to add to the observable,
}
```

To attach files to the alert use the field named **file**.

1.5 Some Background

If you're curious about where ACE came from or the bigger picture of how ACE is meant to be used, the following topics cover some concepts at a high level that should first be understood.

Additionally, John Davison gave a talk on the development of the ACE toolset at BSides Cincinnati in 2015 and covers these same topics. You can watch his presentation here:

1.5.1 Driving Behavior

With the goal set at always detecting advanced attacks and attackers across an organization, you must have detection point coverage across your entire attack surface. This can be challenging in a world of constraints, such as your analysts' time. Analysts cannot be inundated with an unmanageable number of alerts; nor should they be presented with the same alert repeatedly. You need to manage and optimize the volume of alerts presented to analysts. The best way to do this is to get a handle on your False Positive metrics and how those metrics should drive your hunting and tuning behavior.

THE METRIC TO DRIVE: Assume the majority of all alerts are False Positive, then for each alert that is analyzed, how long does it take the analyst to **realize** it is a False Positive?

Why does this metric matter? Because detection is hard and analyst time is highly valuable to a successful security operation.

False Positive Metrics

What is a False Postive? Something that turns out to be nothing? Yes, but more than that, too.

False Positives, False Positive rates, and the average time it takes an analyst to disposition a False Positive are crucial metrics for driving the right security ecosystem.

If your least experienced analyst can't disposition a False Positive in seconds, then it's going to be much harder to both expand and maintain an in-depth coverage of your attack surface. This is, of course, assuming that your security operation is constrained by time, money, and analyst sanity.

Hunting and Tuning!

Hunting is the active process of searching for maliciousness. From hunting, we develop hunts that are meant to detect some specific form of maliciousness. A hunt could be looking for a strange process behavioral pattern, a Yara signature, or just a search for some atomic indicators. When a hunt returns a result, we have a detection and need to create an alert.

Hunts produce True Positives and False Positives. Tuning is the process of telling a hunt not to alert on something we've already determined to be a False Positive. Tune out the False Positives.

Not sure when to hunt and tune? If the detection team can handle 'X' number of alerts in a day, and if 'n' is the number of alerts your tools generate in a day:

- If $n \geq X$ then **tune**.
- If $n < X$ then **hunt** and introduce more alerts for the analysts

Hunt + Tune == Coverage++

With an understanding of your False Positive metrics, hunting and tuning can be used to expand your attack surface coverage.

1.6 Additional Features

The following are additional ACE features that are not necessary to understand when orienting an analyst with ACE or didn't quite fit in other areas of the documentation. If you're here, it's assumed that you're familiar with the content in the *Getting Data into ACE* and *Analyst Orientation* sections.

1.6.1 Events

An event in ACE is a collection of related alerts that require some response activities from your analysts. For example, you can add several phishing alerts that have the same malicious attachment to an event. The event denotes that your analysts have some follow-up work to do on the alerts, such as remediating the email to remove it from the user's inbox or ensuring the user did not click any malicious links or open any malicious files.

We developed a sister project called *Event Sentry* that monitors ACE for events that were created and automatically creates comprehensive wiki write-ups of the event. Other features of Event Sentry include:

- Detects types of malware using built-in and extendable detection modules.
- Detects kill chain phase by determining if a user clicked a link, submitted credentials, or opened a malware sample.
- Extracts indicators from e-mails, sandbox reports, and other artifacts.
- Automatically uploads indicators, samples, and e-mails to CRITs and creates appropriate relationships between them.
- Maintains an event repository containing copies of the ACE alerts and all their artifacts.
- Creates a shareable intel package containing a summary of the event including indicators, malware samples, and emails.

See <https://eventsentry.readthedocs.io/en/latest/> for more information on Event Sentry.

1.6.2 Metrics

ACE's Metrics page can be used to track and display metrics for alert triage operations. Currently, the following tables can be generated:

Alert Quantities Count of alerts by disposition

Hours of Operation Cycle time averages and quantities by the time of day alerts were generated

Alert Cycle Times The average time it took to disposition alerts in business hours

Incidents Summary of incidents (an incident is an event that has progressed beyond DELIVERY)

Events Summary of events

CRITS Indicator Stats Count of indicators by intel source and status

1.7 Administration Guide

1.7.1 Concepts

There are several concepts crucial to understanding how ACE works and how to use ACE. For the analyst, it's important to understand observables, tagging, and dispositioning. The administrator and developer needs to understand those concepts as well, but additionally must understand ACE's dependencies and its engine and modular architecture.

Engines

The ACE system is named after the system's core engine, the Analysis Correlation *Engine*. However, there are additional engines that interface with, utilize, or provide input to the core Analysis Correlation Engine. Below is a table of the currently defined engines:

Engine	Description
ace	The Alert Correlation Engine creates and submits alerts to the Analysis Correlation Engine
carbon_black	Collects binaries and files from CarbonBlack environments and runs them through ACE
brotex_stream	Responsible for analyzing tar files extracted from SMTP and HTTP streams via the Brotex system ¹ . Extracted emails are submitted to the Email Scanning Engine. Extracted HTTP streams are submitted to the HTTP Scanning Engine.
email_scanner	The Email Scanning Engine is configured to fully analyze and scan emails from any available source. There is special support for emails submitted from Office365 (which includes the actual email as an attachment inside the email). The two sources of input for the Email Scanning Engine are the emails parsed out of tar files from the Brotex Engine, which are submitted via local filesystem, and emails collected from the ACE Mailbox Client systems ² , which are submitted via custom SSL connections. Emails that have any alert-able properties are submitted to the Alert Correlation Engine.
http_scanner	Processes and scans individual HTTP requests for malicious content. Alert-able requests are submitted to the Alert Correlation Engine.
cloudphish	Processes, analyzes, crawls, and scans content pulled from received URLs. Maintains a cache of results and a URL whitelisting system. Alert-able URLs are sent to the Alert Correlation Engine. Cloudphish has an API.

Modules

ACE modules automate something that an analyst has previously done manually. These modules do all “the work” on observables; each module knows which types of observables it works with and “knows what to do” with those observables. Modules can be built to do anything that you can automate. Each ACE engine knows which ACE modules to work with, and modules can perform work for many different engines.

Recursive Analysis

With the introduction of observables, engines, and modules, you can begin to understand how ACE performs its recursive analysis and correlation.

For example, given observable type ‘file’, each ACE module that acts on an observable of type file will be called to perform its analysis. From the output of each module’s analysis, ACE will discover and create new observables, which, kicks off more modules to perform analysis. This recursive process will continue until all observables are discovered, analyzed, and correlated, or, until a specified alert correlation timeout is reached. ACE’s default timeout limit for recursive alert analysis is 15 minutes, however, a warning will be logged if alert analysis exceeds five minutes. These values are configurable under ACE’s ‘global’ configuration section.

1.7.2 Turning on Engines

When installed, ACE likely started several engines and modules by default. Almost certainly, the correlation engine was started. You can see below how to stop and start several different engines and modules. If you want to try and start all engines at the same time, the following command will accomplish that:

```
$ /opt/ace/bin/start-ace
```

Correlation Engine

The correlation engine is essential:

```
$ /opt/ace/bin/start-correlation-engine
```

Email Scanner

The email scanning engine will detect any file observable that is compliant with rfc822.

```
$ /opt/ace/bin/start-email-scanning-engine
```

CloudPhish

Make sure **engine_cloudphish** is enabled in `saq.ini`. You may need to add the following enabled variable:

```
[engine_cloudphish]
enabled = yes
```

Also in `saq.ini`, make sure the following config item has this value; unless you know your situation is different. You may have to create this section:

¹ See the Brotex systems on IntegralDefense’s github page: <https://github.com/IntegralDefense>

² The ACE Mailbox Client is open sourced at <https://github.com/IntegralDefense/amc.git>

```
[analysis_module_cloudphish]
cloudphish.1 = https://localhost/ace/cloudphish
```

The CloudPhish engine depends on the CrawlPhish analysis module. So make sure the **analysis_module_crawlphish** is turned on in `saq.ini`. You may have to create this section:

```
[analysis_module_crawlphish]
enabled = yes
```

Next, make sure the following three files exist. Example content is given for each file. First, `/opt/ace/etc/crawlphish.whitelist`:

```
# url shorteners and more
anonfile.xyz
bit.ly
goo.gl
ow.ly
is.gd
dd.tt
dropbox.com
tinyurl.com
zip.net
drive.google.com
wetransfer.com
hyperurl.co
ldrv.ms
onedrive.live.com
amazonaws.com
```

Second, `etc/crawlphish.path_regex::`

```
# possible file extensions for trojans
\.
→ (pdf|zip|scr|js|cmd|bat|ps1|doc|docx|xls|xlsx|ppt|pptx|exe|vbs|vbe|jse|wsh|cpl|rar|ace|hta)
→ $
```

Finally, `etc/crawlphish.blacklist`:

```
# ignore loopback
127.0.0.1
# RFC 1918
10.0.0.0/8
172.16.0.0/12
192.168.0.0/16
# put more domains and IPs you want to avoid
```

Finally, everything is ready to turn on the cloudphish engine:

```
$ bin/start-cloudphish
```

1.7.3 Enabling Modules

Yara Scanner

First, make sure the **analysis_module_yara_scanner_v3_4** section in `/opt/ace/etc/saq.ini` is enabled. Then create a `/opt/signatures` directory:

```
$ mkdir /opt/signatures
$ cd /opt/signatures
```

Now place your yara signature directories in */opt/signatures/<your yara directories>*.

Create a symlink for ACE to find your signatures:

```
$ ln -s /opt/signatures $SAQ_HOME/etc/yara
```

Start the yara module:

```
$ /opt/ace/bin/start-yss
```

Live Renderer

The live browser rendering module will try to render a png image of any html file it's given. This can be particularly helpful for viewing email html content. Keep security in-mind when implementing this module.

To configure the module, execute the following commands. NOTE: The following instructions explain how to set up the renderer on localhost, but you can set up the rendered on a dedicated server as well.

Create a user named “cybersecurity”:

```
$ sudo adduser cybersecurity
```

Generate a ssh key as the ace user:

```
$ ssh-keygen -t rsa -b 4096
```

Add this entry to your ace ssh config:

```
$ cd /home/ace
$ vim .ssh/config

Host render-server
  HostName localhost
  port 22
  User cybersecurity
  IdentityFile /home/ace/.ssh/id_rsa
```

Set up the cybersecurity account:

```
$ sudo su - cybersecurity
$ cd && mkdir .ssh && mkdir tmp
$ cat /home/ace/.ssh/id_rsa.pub >> .ssh/authorized_keys
$ ln -s /opt/ace/render render
$ exit
```

Add localhost as a known ssh host for the ace user:

```
$ ssh-keyscan -H localhost >> .ssh/known_hosts
```

Run the install script:

```
$ cd /opt/ace/render/ && ./install
```

Download the most recent Chrome driver from <https://sites.google.com/a/chromium.org/chromedriver/downloads>:

```
$ cd /opt/ace/render
$ wget https://chromedriver.storage.googleapis.com/<version number goes here>/
  ↳chromedriver_linux64.zip
$ unzip chromedriver_linux64.zi
```

Finally, make sure the following (at a minimum) is in your `saq.ini` file:

```
[analysis_module_live_browser_analyzer]
remote_server = render-server
enabled = yes
```

Now, restart the correlation engine and render away.

1.8 Development Guide

1.9 Developer README

This document explains the reasons behind some of the stranger design decisions made for this project.

1.9.1 SAQ = ACE

When the project first started we called it the Simple Alert Queue (SAQ). It was later renamed to the Analysis Correlation Engine (ACE). There are still a lot of references to SAQ left, including the name of the core library (`import saq`) and the `SAQ_HOME` environment variable.

1.9.2 Everything was initially command line driven.

The original UI of the project was CLI. So there's still a lot of that left. Most of what you can do can also be done via the command line, including full analysis of observables.

Along those lines, it was also meant to be able to be executed from any directory. This is probably no longer true, but there are a number of times where the code assumes it is running in some other directory.

1.9.3 This was an internal project.

There's a number of basic things that you would expect would exist that don't. For example, there's no way to manage users from the GUI. It must be done from the command line. And even then, there's no support to delete a user. We didn't have any turnover for 5 years so this was never a requirement.

And the along those lines there's little effort put into account security internally. There are no "roles" or "administrators".

1.9.4 The database came later.

Very little of the analysis data is stored in the database.

From the beginning of the project I wanted the data to be stored in a schema-less JSON structure on the filesystem. This would allow analysts to simply `grep` the files for whatever they were looking for. I (reluctantly) looked at MongoDB as a way to index the data and speed up the searches. This was quickly abandoned (it was slowing down development for various reasons.) Later when the GUI was added to the project we started storing data in MySQL.

I knew that we would be modifying this system a lot. So trying to create a database schema that encompassed everything we would ever want to do was not realistic. Making major changes to large database schemas is no easy task.

Today the database is used to manage the workload of the collectors and engines, and to provide the GUI (and API) for the analysts. The `data.json` JSON files that hold the results of the analysis are actually the official records of the analysis. The database is kept in sync with these files.

At some point it would make sense to index these JSON files in a system like Elasticsearch.

1.9.5 Unit testing.

My one regret with this project was not creating unit tests as I went. I didn't start adding unit tests until we were ~4 years into the project. Unit test coverage is not what it should be, but I would expect that to improve over time.

1.9.6 Final words.

I think it's worth noting that this project was created to enable and improve our analysts. We were not designing a *product*. We were also moving as quickly as we saw threat actors change tactics. As soon as we saw a new technique being used, we would quickly implement a feature to ACE that would allow us to detect that. So there's a number of places where the code looks hastily thrown together.

Hopefully this file helps to explain some of the oddness you may see in the code.

CHAPTER 2

Indices and tables

- `genindex`
- `modindex`
- `search`