OX Abuse Shield

Release Notes for Release 2.2.0

2019-11-07

Copyright notice

1 General Information

Open-Xchange is pleased to announce the release of OX Abuse Shield v2.2.0.

OX Abuse Shield provides abuse-prevention for Web Applications (including Webmail), POP, andIMAP. It is integrated with both OX App Suite and Dovecot Pro to prevent login and authenticationabuse as well as protecting against brute-force attacks.

The goal of OX Abuse Shield is to detect brute forcing of passwords across many servers, service-sand instances, as well as enforce policy for authentication and authorization. In order to support-the real world, brute force detection policy can be tailored to deal with ”bulk, but legitimate” usersof your service, as well as botnet-wide slow-scans of passwords.

The new OX Abuse Shield v2.2.0 provides the following main improvements and new features

- Lookup individual GeoIP2 values (Unsigned Integer, String, Double/Float and Boolean)
- Add session_id to wforce LoginTuple
- The Report API is now deployable via proper packaging, systemd and gunicorn
- Custom GET Endpoints supporting non-JSON return values
- New Kibana Reports and Dashboard to view effectiveness of wforce policy
- New ”type” field added to built-in webhooks
- Built-in whitelists added in addition to built-in blacklists
- New checkBlacklist and checkWhitelist functions in Lua
- Built-in black/whitelisting can be disabled and checked from Lua instead
- Thread names support
- Built-In Blacklist and Whitelist return messages are configurable
- Support TCP Keepalive

The new OX Abuse Shield v2.2.0 provides the following bugfixes

- Fix typo in wforce.conf.example blackistLogin->blacklistLogin
- Python3 support throughout wforce (regression tests and deployment)
- Uses ”bytes” instead of ”string” in replication, which avoids protobuf errors for non-UTF-8 StatsDB keys or values.
- Remove ctpl from wforce
- Sibling threads now use explicit std::queue instead of ctpl
- Only connect to TCP siblings when necessary (not on startup) to prevent delays on startup.
- Updated logstash.conf file for ELK integration

The new wforce-policy v2.2.0 provides the following main improvements and new features

- TestMode for testing wforce in production
- delayedWhitelisting feature to see whether whitelisted IPs and users are actually triggering policies

- Default to sending both allow *and* report commands to Elasticsearch
- Add support for pdns-builder to build packages in-repo
- Use custom logging to log the ”log” action and whitelisted IPs. Otherwise these would not get logged unless verbose logging was enabled.
- Lua-based checks of the built-in blacklists (for use where built-in blacklist checking is disabled in wforce).
- Hooks to create Custom Policies and Fields

The new wforce-policy v2.2.0 provides the following bugfixes

- Fix issue where empty device_id or unparseable device_id was treated as new device in trackalert
- Fix bug where whitelisting did not always occur
- Fix typo where mobileapi was mobileapp
- Fix device handling in smtp.lua which was completely broken
- Fix lua exception when policy_action is not set, and log error message
- Fix invalid variable name bug in wforce.lua allow function

What’s New in General

Open-Xchange now provides more detailed overviews, data sheet and product guide, relating to new product major release. These can be found at https://www.open-xchange.com/portfolio/whats-new/

Download and Installation

For further details about OX Abuse Shield installation, mandatory and optional packages, policies, please refer to the documentation provided:http://oxpedia.org/wiki/index.php?title=AppSuite:OX_Abuse_Shield

Lookup Individual GeoIP2 Values

Previously the GeoIP2 support was limited to retrieving City and County DB information, with a fixed set of fields returned. However this meant that other DBs (e.g. ISPs, Anonymizers etc.) could not be queried, as well as additional fields in City or Country DBs. Now there are four new Lua functions to query arbitrary fields in the Maxmind DBs:

- lookupStringValue
- lookupDoubleValue
- lookupUIntValue
- lookupBoolValue

For example:

```
local citydb = getGeoIP2DB("City")
local city_name = citydb:lookupStringValue(newCA(ip_address), {"city", "names", "en"})
local city_lat = citydb:lookupDoubleValue(newCA(ip_address), {"location", "latitude"})
local city_long = citydb:lookupDoubleValue(newCA(ip_address), {"location", "longitude"})
local accuracy = citydb:lookupUIntValue(newCA(ip_address), {"location", "accuracy_radius"})
local eu = citydb:lookupBoolValue(newCA(ip_address), {"country", "is_in_european_union"})
```

Add session_id to wforce LoginTuple

It can be useful for wforce to recognise when multiple logins belong to the same session. Therefore a new field ”session_id” is added to the LoginTuple table, which allows a session id to be passed to wforce. This field is optional, so if empty it should be ignored. For example:

```
local session_id = lt.session_id
```

Deployable Report API

Previously the Report API was supplied as an experimental feature, which was supplied as a Flask application, but without support for running in production, packaging etc.

Now the Report API is fully deployable; it is packaged in a new package ”wforce-report-api”, and can be started/stopped via systemd. For example:

```
systemctl enable wforce-report-api
systemctl start wforce-report-api
```

The Report API is still a Flash application, which runs under gunicorn, and the configuration and deployment settings are configurable.

Report API configuration is via:

```
/etc/wforce-report-api/wforce-report-api-instance.conf
```

Gunicorn configuration is via:

/etc/wforce-report-api/wforce-report-api-web.conf

It is recommended to run Gunicorn behind an nginx proxy in a production deployment.

Custom GET Endpoints

The existing Custom Endpoint functionality is based only on POST commands, and requires return values to be json-encoded. This presents problems with more limited clients such as firewalls or other network equipment, which only support HTTP GET, and cannot parse json-encoded return values.

The Custom GET endpoints solve this issue by allowing endpoints to be created which are based on HTTP GET, and which return data using the text/plain content type.

Custom GET endpoints do not take any parameters, so there is no way to pass data to the endpoints.

For example the following function uses the new getIPBlacklist() functionality to return a text version of the IP blacklist:

```
function returnTextBlacklist()
local ipbl = getIPBlacklist()
local ret_table = {}
for i,j in pairs(ipbl)
do
for k,v in pairs(j)
do
if k == "ip"
```

```
then
table.insert(ret_table, v)
end
end
end
local s = table.concat(ret_table, "\n") .. "\n"
unblacklistIP(newCA("1.2.3.4"))
return s end
setCustomGetEndpoint("textBlacklist", returnTextBlacklist)
```

New Kibana Reports and Dashboard

New Kibana Reports and Dashboard have been added to the kibana_saved_objects.json file. These reports are based on ”allow” webhooks sent to elasticsearch.

New ”type” field added to built-in webhooks

The built-in webhooks add a ”type” field to the webhook json to make it easier to search for specific webhook types in elasticsearch. The type field can have the following values:

- wforce_report
- wforce_allow
- wforce_reset
- wforce_expireblwl
- wforce_addblwl
- wforce_delblwl

Built-in Whitelists

In addition to the built-in blacklists there are now built-in whitelists. Entries can be added and deleted from the built-in whitelists using either the REST API or using Lua commands.

For example using the REST API:

```
curl -XPOST --data '{ "ip":"1.2.3.4" }' -u user:pass http://localhost:8084/?command=addWLEntry
```

For example using Lua:

```
whitelistIP(lt.remote, 3600, "This is the reason why it is blacklisted")
```

New checkBlacklist and checkWhitelist functions in Lua

New functions to check the built-in black and whitelists are available from Lua:

- checkBlacklistIP
- checkBlacklistLogin
- checkBlacklistIPLogin
- checkWhitelistIP

- checkWhitelistLogin
- checkWhitelistIPLogin

See the wforce.conf man page for more details.

Built-in Black/Whitelisting can be disabled

The built-in black/whitelisting functionality can be disabled so that the checks can instead be performed from Lua. This is achieved with the following Lua commands:

```
disableBuiltinBlacklists()
disableBuiltinWhitelists()
```

Thread Names Support

Every thread type in wforce is now named, so that top -H for example will show individual thread names. This can be very useful for diagnosing when particular threads are CPU-bound and could benefit from increasing the size of the thread pool for example.

Built-In Blacklist return messages are configurable

The following new functions enable the return messages for built-in blacklists to be configured:

- setBlacklistIPRetMsg
- setBlacklistLoginRetMsg
- setBlackistIPLoginRetMsg

See the wforce.conf man page for more details.

Support TCP Keepalive

The wforce daemon previously did not enable TCP keepalive on accepted sockets. The TCP keepalive socketoption is now enabled for all sockets.

TestMode for testing wforce in production

The top-level configuration parameter ”testMode” can be used to put wforce into production without causing any tarpit or reject actions to be returned. Logging is the same as if testMode was disabled, however the key ”test_mode=1” will be set in the ret_attrs object.

delayedWhitelisting Feature

The ”delayedWhitelisting” feature allows the full policy to be run for whitelisted IPs. This is used to examine the bahviour of whitelisted IPs. See the wforce_policy manpage for more details.

Default to sending both allow *and* report commands to Elasticsearch

Now when elasticsearch support is enabled, both allow and report command details will be sent to elasticsearch by default. This can be modified by configuring the ”commands” parameter of the elasticsearch config parameter. See the wforce_policy manpage for more details.

Use custom logging to log the ”log” action and whitelisted IPs

Previously for allow commands that triggered the ”log” action, or were whitelisted, the allow log would only be reported if verbose logging was enabled (this is because the allow log only logs non-zero return values by default to minimize the amount of logging). Now both ’log’ actions and whitelisted IPs/users will be logged using custom lua logging.

Lua-based checks of the built-in blacklists

The built-in blacklists can now be disabled (see wforce.conf manpage), and instead the checks can be performed from Lua. The default policy now includes support for this with the ”checkBuiltIn-Blacklist” top-level configuration parameter. See the wforce_policy manpage for more details.

Hooks to create Custom Policies and Fields

New functions to register custom policies, which consist of the following:

- Create a new statsDB field which can be used to track new statistics
- Create a new policy that uses a threshold based on a specific statsDB field
- Create a completely custom policy using a Lua function

These new functions are all documented in wforce-policy.conf.

Fix issue where empty device_id or unparseable device_id was treated as new device in trackalert

An empty or unparseable device_id field would cause trackalert policy to record a new device and send an alert. Since empty device information is not saved, this would generate a new device alert everytime such a login was recorded. Empty or unparseable devices no longer register as new devices.

Fix bug where whitelisting did not always occur

A typo in wforce.lua meant that sometime whitelisting would not occur when it should. This has been fixed.

Fix typo where mobileapi was mobileapp

The OX mobile applications use a special User-Agent string which is recorded by wforce as ”mobileapi”. A typo meant that mobileapp was used in some cases, causing trackalert new device detection to fail in certain cases.

Fix device handling in smtp.lua

The device handling for SMTP alerts was broken such that device information did not get inserted into SMTP messages. This has been fixed.

Fix lua exception when policy_action is not set, and log error message

If a wforce policy was created without a policy_action (specifically tarpit or any blacklist action), then this would generate a Lua exception and the allow call would return without any return values. Now this exception is detected, an error log message is generated, and a default value of 1 is assigned to the policy_action.

Fix invalid variable name bug in wforce.lua allow function

An invalid variable name ”rc” was used instead of ”retval” when checking the results of some policy functions.

2 Shipped Product and Version

OX Abuse Shield v2.2.0-rev1

Find more information about product versions and releases at http://oxpedia.org/wiki/index.php?title=AppSuite:Versioning_and_Numbering and http://documentation.open-xchange.com/.