Stay Open.

Open-Xchange Abuse Shield

Major Release v2.2.1

Product Guide

1 OX Abuse Shield

1.1 Intention of this document

This document provides an overview of the improvements and other changes that come with this major release of OX Abuse Shield, v2.2.

The purpose of this document is to inform Open-Xchange customers and partners about the major changes that have been made in this release.

1.2 OX Abuse Shield in General

OX Abuse works with both Dovecot Pro and OX App Suite, or any third-party software via a REST API, as a component to protect against login/authentication abuse.

OX Abuse Shield runs on a cluster of servers and integrates with both OX App Suite and Dovecot to detect abuse, brute force attacks and also to enforce common authentication/authorization policies across the platform.

1.3 Key Features

- Replicated/clustered architecture – Login reports are shared between all the servers in a cluster so there is a single view of abuse
- Scriptable Policy Language – Using the Lua language, the functionality of the daemon can be extended to record and protect against a large variety of abusive behavior, as well as implement specific customer policies.
- DNS Lookup Feature – For looking up IPs or domains in blacklists
- GeoIP Lookup Feature – GeoIP lookups can be made and incorporated into policy decisions.
- Rate limiting and Tarpitting – Extremely flexible, these can be enabled and enforced based on IP address, login name, geoip location, time windows, etc.

- Flexible In-Memory Statistics Database – A versatile and extensible in-memory database is used to store statistics information about abuse over time periods from a few minutes to many hours.
- Integration with Customer Authentication/Authorization Systems – Customers can use the open HTTP REST API to benefit from the protection of the anti-abuse daemon in their own authentication/authorization systems.
- Admin Console – To retrieve statistics and query server state
- Persistent Replicated Blacklist – Configurable via a REST API or the Lua policy engine, supports auto-expiry of entries, replication between all cluster nodes, and optionally uses a Redis DB for persistence.
- Webhooks – Integrate Anti-Abuse Shield with other systems using webhooks to send events.

A more detailed overview of OX Abuse Shield can be found at the

https://software.open-xchange.com/products/weakforced/doc/OX_Whitepaper_OX_Abuse_Shield_2_2_1.pdf

1.4 The Goal of OX Abuse Shield

The goal of OX Abuse Shield is to detect brute forcing of passwords across many servers, services and 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" users of your service, as well as botnet-wide slow-scans of passwords.

Here is how it works:

- Report successful logins via JSON http-api
- Report unsuccessful logins via JSON http-api
- Query if a login should be allowed to proceed, should be delayed, or ignored via JSON http-api

- OX App Suite and Dovecot's POP/IMAP server are pre-integrated with OX Abuse Shield. Other software can be integrated easily using the REST API

2 Key benefits of OX Abuse Shield v2.2

Open-Xchange is pleased to announce the release of OX Abuse Shield v2.2.

Keeping in line with Open-Xchange's security strategy OX Abuse Shield v2.2 contains many enhancements designed specifically for end-users and end-user usability: usability that stretches beyond the OX App Suite user base.

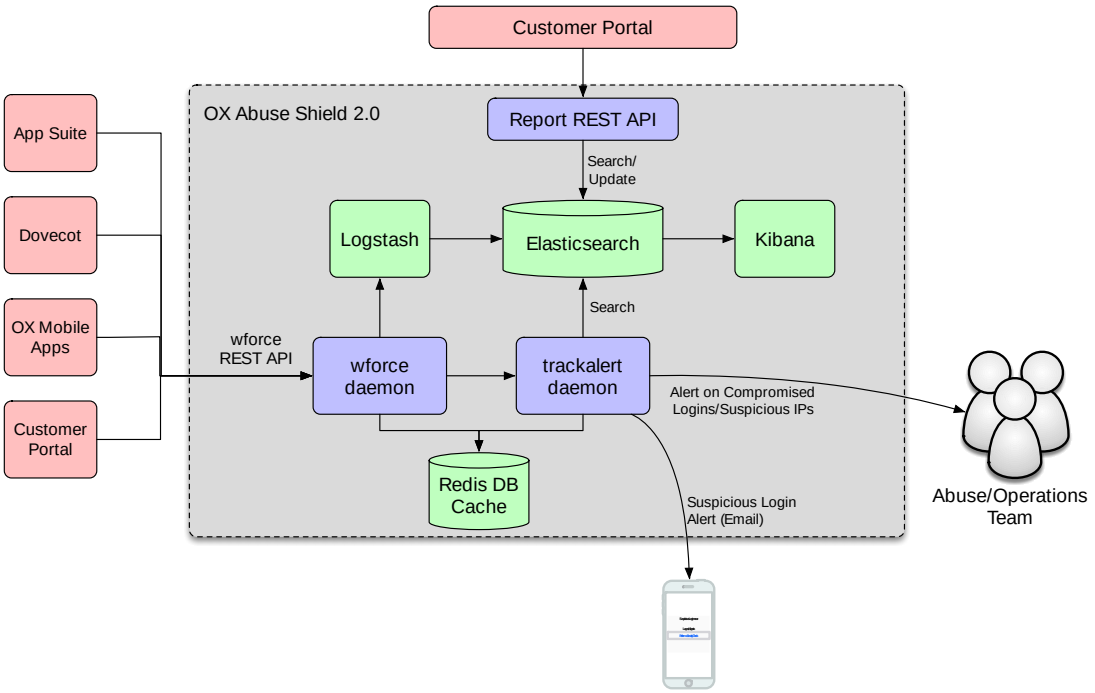
2.1 New Features

- 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
- Hooks to create Custom Policies and Fields

3 In Detail – Benefits and new Enhancements

3.1 Overview

For an overview of the existing and new components and how they fit together, see the following diagram:



3.2 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

3.3 Add session_id to wforce LoginTuple

It can be useful for wforce to recognize 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

3.4 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.

3.5 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.

3.6 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.

3.7 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

3.8 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.

3.9 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.

3.10 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.

3.11 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.

3.12 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.

3.13 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.

3.14 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.