SYHUNT HYBRID: LUA API

The information in this document applies to **version 6.9.12** of Syhunt Hybrid.

INTRODUCTION

Syhunt Hybrid Platinum comes with a simple, easy-to-use Lua API that allows to launch dynamic and source code scans, get status, report and log of a launched scan session, and more.

LOADING THE SYHUNT LIBRARY

Before using Syhunt's Lua API, you must load the Syhunt library:

```
-- From GUI app
require "SyMini"
-- From Console app
require "SyMini.Console"
```

LAUNCH A DYNAMIC SCAN

```
local res = symini.scantask_launch({
  sessionname = "Test",
  starturl = "http://127.0.0.1",
  huntmethod = "appscan",
  reporttemplate = "Complete"
})
if res.result == true then
  print(res.sessionname)
end
```

Keys explained:

- `sessionname` (optional) - must contain an unique, alphanumeric session name. If omitted, a random one will be generated.

- `starturl` (required) - the target URL
- `huntmethod` (optional) - a valid scan method name. For a list of valid hunt methods, see Differences between hunt methods.
- `reporttemplate` (optional) - a valid template report name (Standard, Comparison, Compliance or Complete)
- `reportgl` (optional) - If true, returns a JSON output using GitLab format.
- `timelimit` (optional) - Sets the maximum scan time limit (default: no limit). If the time is reached, the scan is aborted. Examples: 1d, 3h, 2h30m, 50m

Return table keys:

- `pid` - An unique process ID associated with the active scan,
- `result` - A true result means the scan has been launched. Otherwise a false means it was not possible to start the scan.
- `resultstr` - If the result was false, this will contain an error description.
- `sessionname` - The session name associated with the scan. If you omitted the sessionname key within the request body, this key will contain a randomly generated session name. You must use this session name as part of subsequent requests when getting scan results.
- `sessiontype` - The type of the session (dynamic or code scan)

LAUNCH A SOURCE CODE SCAN

```
-- Scan a local directory
local res = symini.scantask_launch({
  sessionname = "Test",
  sourcetarget = "P:\\Private\\MyWebApp\\",
  huntmethod = "normal",
  reporttemplate = "Complete"
})

-- Scan a project URL
local res = symini.scantask_launch({
  sessionname = "Test",
  sourcetarget = "https://github.com/drnic/php-helloworld.git",
  sourcebranch = "master",
  huntmethod = "normal",
  reporttemplate = "Complete"
})

if res.result == true then
  print(res.sessionname)
end
```

Keys explained:

- `sessionname` (optional) - must contain an unique, alphanumeric session name. If omitted, a random one will be generated.
- `sourcetarget` (required) - a local target directory or a GIT URL
- `huntmethod` (optional) - a valid scan method name. For a list of valid hunt methods, see Differences between hunt methods.
- `reporttemplate` (optional) - a valid template report name (Standard, Comparison, Compliance or Complete)
- `reportgl` (optional) - If true, returns a JSON output using GitLab format.
- `timelimit` (optional) - Sets the maximum scan time limit (default: no limit). If the time is reached, the scan is aborted. Examples: 1d, 3h, 2h30m, 50m

Return table keys:

- `pid` - An unique process ID associated with the active scan,
- `result` - A true result means the scan has been launched. Otherwise a false means it was not possible to start the scan.
- `resultstr` - If the result was false, this will contain an error description.
- `sessionname` - The session name associated with the scan. If you omitted the sessionname key within the request body, this key will contain a randomly generated session name. You must use this session name as part of subsequent requests when getting scan results.
- `sessiontype` - The type of the session (dynamic or code scan)

GET RESULTS (STATUS, REPORT OR LOG)

GET STATUS

```
local res = symini.scantask_getresults({
  sessionname = "Test",
  resulttype = "status"
})
```

Keys explained:

- `sessionname` (required) - the name of a scan session you want to obtain results
- `resulttype` (required) - the type of result you expect (can be status, report_xml or session_log)

Return table keys:

- `report_xml_available` - True if a XML report is already available, false otherwise.
- `report_json_available` - True if a JSON report is already available, false otherwise.
- `session_log_available` - True if a scan log is already available, false otherwise.
- `status` - the scan session status (Scanning or Completed, when it has finished)

GET REPORT

```
local res = symini.scantask_getresults({
  sessionname = "Test",
  resulttype = "report_xml"
})

-- Alternatively you can obtain the JSON report
local res = symini.scantask_getresults({
  sessionname = "Test",
  resulttype = "report_json"
})
```

GET LOG

```
local res = symini.scantask_getresults({
  sessionname = "Test",
  resulttype = "session_log"
})
```

The return will contain the scan log in text format.

GENERATE A SCAN REPORT OR EXPORT

```
local res = symini.genreport({
    sessionname = 'mysessionname',
    outfilename = 'myreport.pdf',
    template = 'Standard'
})
```

Keys explained:

- `sessionname` (required) - the name of a scan session you want to generate a report
- `outfilename` (required) - the output filename of the report
- `template` (optional) - the template name (Default: Standard)
- `passfailcond` (optional) - a pass/fail condition.

Return table keys:

- `outfilename` - The full output filename of the report.
- `result` - True if the report was generated, false otherwise.
- `resultstr` - Status of the report generation.
- `passfail_result` - True if passed the specified condition, false otherwise.
- `passfail_status` - Status of the pass/fail condition.
- `passfail_resultstr` - Reason of the pass/fail condition.

GET SESSION DETAILS

```
local res = symini.getsessiondetails('mysessionname')
```

Return table keys:

- `huntmethod` - Hunt method
- `datetime` - Date/Time of the scan
- `duration` - Duration of the scan
- `targets` - Target(s) of the scan
- `targetdesc` - Description of the target
- `sourcedir` - Target source code directory (if any)
- `targetfile` - Target file (if any)
- `targeturl` - Target URL (if any)
- `vulncount` - Total of vulnerabilities
- `attackcount` - Total of attacks (if this is a web server log scan)
- `ports` - Target ports (if any)
- `status` - Status of the session
- `resultsdesc` - Description of the results
- `launcher` - The name of the tool that launched the session

EMAIL A REPORT

```
local res = symini.emailreport({
   tracker = 'myemailtrackername',
   filename = 'myreport.pdf',
   subject = 'My Test Report'
})
```

Keys explained:

- `tracker` (required) - the name of a previously enabled email issue tracker.
- `filename` (required) - the full filename of the report to be sent.
- `subject` (optional) - the subject of the email. Default: Syhunt Scanner Report

Return table keys:

- `result` - True if the operation was successful, false otherwise.
- `resultstr` - Status of the attempt containing an error message (if any).
- `resultdbg` - Debug log of the operation.

UPDATE SYHUNT PREFERENCES

- **symini.prefs_set** (key, value [, targeturl]) - Sets a permanent preference (global or target-specific)
- **symini.prefs_get** (key, value [, targeturl]) - Gets a permanent preference (global or target-specific)

A list of available preference IDs is available here.

```
-- Update a site preference
local res = symini.prefs_set('enabled', true, 'http://127.0.0.1')
-- Update a global preference
local res = symini.prefs_set('hybrid.report.company.logo.url', 'https://www.mydomain.com/mylogo.png')
if (res.result == true) then
  print('Preference updated.')
end
```

Return table keys:

- `result` - True if the operation was successful, false otherwise.
- `resultstr` - Status of the attempt containing an error message (if any).
- `value` - Value of preference (only returned by prefs_get).

ADDITIONAL API FUNCTIONS

- **symini.info.version** - Contains the Syhunt version.
- **symini.getsessionname** ( ) - Returns a random session name that can be used when launching a scan.
- **symini.getsessioncomparison** (sessionname1, sessionname2) - Returns a comparison of two sessions.
- **symini.getsessionlist** ([maxdays]) - Returns the list of past sessions.
- **symini.runcmd** (cmdname) - Executes a command. Available commands include:
  - **stop** - Stops all running processes from this Syhunt installation
  - **clearinc** - Clears the incremental cache (if any)
  - **clearpref** - Clears the global preferences
  - **clearsite** - Clears the site preferences (if any)
  - **cleartrack** - Clears any added issue trackers

The scantools repository includes additional Syhunt API usage examples using objects.

For additional product documentation, visit **syhunt.com/docs**

SYHUNT®