
Breach Remediation

Windows Administrator Guide

Version 4.3.17

23 October 2024

Notices

Our products and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. You may copy and use this document for your internal reference purposes only.

This document is provided “as-is.” The information contained in this document is subject to change without notice and is not warranted to be error-free. If you find any errors, we would appreciate your comments; please report them to us in writing.

Third Party Project Usage

Our software is made possible thanks in part to many open-source and third-party projects. A requirement of many of these projects is that credit is given where credit is due. Information about each third-party/open-source project used in our software – as well as licenses for each – are available for viewing here:

<https://support.threatdown.com/hc/en-us/articles/4414986433683>

Sample Code in Documentation

The sample code described herein is provided on an "as is" basis, without warranty of any kind, to the fullest extent permitted by law. We do not warrant or guarantee the individual success developers may have in implementing the sample code on their development platforms. You are solely responsible for testing and maintaining all scripts.

We do not warrant, guarantee or make any representations regarding the use, results of use, accuracy, timeliness or completeness of any data or information relating to the sample code. We disclaim all warranties, express or implied, and in particular, disclaims all warranties of merchantability, fitness for a particular purpose, and warranties related to the code, or any service or software related there to.

The Protection Strategy

Our products incorporate several prevention features which utilize a layered defense strategy to protect you against malware threats which you face daily. Each layer is designed to disrupt the attack chain at a different stage. While all products are highly effective in dealing with attacks that are becoming all too commonplace, our protection capabilities are most effective when you take advantage of the full product suite, allowing each prevention layer to do the job they are best suited for.

It's your data. Protect it wisely!

Table of Contents

Introduction.....	1
What's New	1
Key Features	1
External Access Requirements	2
System Requirements	2
Using Breach Remediation.....	3
License Key Status.....	3
Getting Started	3
Interactions with Anti-Rootkit Scanning	4
Remediation Now or Later?.....	5
Remediation Scan	5
Diagnostic Scan	5
Selective Remediation.....	6
Scan Summary	9
Excluding Items from Scanning	9
Restoring Items from Quarantine	11
Command Line Parameters	13
Conventions	13
Command Line Overview.....	13
Command Line Reference.....	14
register	14
update	14
version.....	14
scan	15
errorout.....	17
quarantine.....	18
settings.....	19
Scan Log.....	22
<i>header</i> Section	22
applicationVersion	22
clientID	22
clientType	22
componentsUpdatePackageVersion.....	22
cpu.....	22
dbSDKUpdatePackageVersion.....	22
detectionDateTime	22
fileSystem	22
id.....	22
isUserAdmin.....	23
licenseState	23

linkagePhaseComplete23

Table of Contents

loggedOnUserName23

machineID23

os23

schemaVersion.....23

sourceDetails Section..... 23

 aggressiveMode23

 clientMetadata23

 objectsScanned.....23

 scanOnlineStatus.....23

 shurikenEnabled23

 scanState23

 threatsDetected.....23

 scanStartTime24

 scanEndTime.....24

 scanResult.....24

scanOptions Section 24

 pumHandling.....24

 pupHandling.....24

 scanArchives24

 scanFilesystem.....24

 scanMemoryObjects.....25

 scanPUMs.....25

 scanPUPs25

 scanRootkits.....25

 scanStartupAndRegistry25

 scanType.....25

 useHeuristics.....25

threats Section 25

 ruleID25

 rulesVersion25

 threatID25

 threatName.....25

mainTrace Section 26

 cleanAction26

 cleanContext.....26

 cleanResult.....26

 cleanResultErrorCode.....26

 cleanTime26

 generatedByPostCleanupAction26

 id.....26

 linkType.....26

 objectMD5.....26

 objectPath26

 objectSHA256.....26

objectType.....26

Table of Contents

SuggestedAction26

linkedTraces Section 27

 ruleID27

 rulesVersion27

 threatID27

 threatName.....27

Sample Scan Progress File 28

Event Logging to syslog 29

 Construction of a Log Entry..... 29

 Mapping Fields to CEF Format 29

Log Events..... 30

 1000 – ScanStartEvent.....30

 1001 – DetectionEvent.....30

 1002 – RemovalEvent30

 1003 – ScanEndEvent.....31

 1004 – RestoreStartEvent31

 1005 – RestoreEvent31

 1006 – RestoreEndEvent.....31

 1007 – RemoveLastScanEvent.....31

 1008 – DbUpdateEvent32

 1009 – CustomDbUpdateEvent32

Creating Custom Rules..... 32

 Custom Hash Rule.....34

 Custom File Rule.....34

 Custom Folder Rule35

 Custom Registry Key Rule.....35

 Custom Registry Value Rule.....35

Program Status Codes36

Further Reading36

Appendix A: Demo Examples..... 36

Introduction

Breach Remediation is designed to allow business users to detect and remove malware from endpoints. It is built upon the power of our flagship anti-malware client, which allows *Breach Remediation* to run in environments which often render other anti-malware applications helpless.

Implementation in a portable form provides increased flexibility for IT staff to quickly and easily deploy the client, use it to remediate threats, gather logs, and continue with their daily tasks – all without a large investment in time or resources.

What's New

The following changes have been made to *Breach Remediation* in this version.

- Added a command to change the licensing server.
- Changed the default licensing server.

Key Features

Breach Remediation offers the following key features:

- Combines signature-based, signature-less, and intelligent heuristics to analyze potential threats, even those designed to evade signatures
- Selective remediation capability
- Remediation of earlier scan results without requiring a second scan
- Four different types of scans to analyze your endpoint for malware threats, regardless of whether they are based in memory, file system or registry
- Ability to perform full scans for all local drives
- Ability to utilize threat definition updates, assuring that even the newest threats can be detected
- Ability to quarantine detected threats, and to restore if needed
- Ability to deploy client to endpoints using your preferred methods
- Ability to exclude several object types from scanning
- Command line capabilities allow IT staff to modify certain program configuration settings, execute scans, and gather logs through integration with customer-supplied scripts, batch files, and group policy updates
- Client leaves no lasting footprint on endpoint
- CEF-compatible event logging
- Ability to use the program in *scan only* or *scan and remediate* mode.
- Ability to configure XML and JSON Output. To enable or disable XML Output, in addition to JSON, add the following system environment variable **MBBR_XML** as follows:
 - MBBR_XML = 1 – XML enabled (XML and JSON scan & clean output files)
 - MBBR_XML = 0 – XML disabled (JSON only scan & clean output files). (Default)
- Ability to configure Trace Logging. To enable or disable more detailed logging and troubleshooting output in the MBBR-ERROUT.TXT logfile, add the following system environment variable **MBBR_TRACE** as follows:
 - MBBR_TRACE = 1 – Trace Logging enabled
 - MBBR_TRACE = 0 – TRACE Logging disabled. (Default)

External Access Requirements

If your company's Internet access is controlled by a firewall or other access-limiting device, you must grant access for *Breach Remediation* to reach our services. The services which must be accessible are:

- <https://cdn.mwbsys.com> Port 443 outbound
- <https://hubble.mb-cosmos.com> Port 443 outbound
- <https://keystone.mwbsys.com> Port 443 outbound
- <https://sirius.mwbsys.com> Port 443 outbound
- <https://telemetry.malwarebytes.com> Port 443 outbound
- <https://blitz.mb-cosmos.com> Port 443 outbound
- <https://brix.threatdown.com> Port 443 outbound

PLEASE NOTE: None of the URLs listed above are guaranteed to be able to be accessed via standard networking tools or browser interactions. This helps to prevent DOS attacks.

System Requirements

Following are minimum requirements for an endpoint on which *Breach Remediation* may be installed. Please note that these requirements do not include other functionality that the endpoint is responsible for.

- **Operating Systems:**
 - Windows 11 (64-bit)
 - Windows 10 (32/64-bit)
 - Windows 8.1 (32/64-bit)
 - Windows 8 (32/64-bit)
 - Windows 7 (32/64-bit) (Service Pack 1 or later)
 - Windows Server 2008 R2 (64-bit)
 - Windows Server 2012/2012 R2 (64-bit only)
 - Windows Small Business Server 2011 (64-bit only)
 - Windows Server 2016 (64-bit)
 - Windows Server 2019 (64-bit)
 - Windows Server 2022 (64-bit)

PLEASE NOTE: Windows servers using the Server Core Installation process are specifically excluded. KB4474419 and KB4490628 are required for Windows 7 and Windows Server 2008 R2.

- **CPU:** 800 MHz or faster
- **RAM:** 512 MB (1 GB or more recommended)
- **Free Disk Space:** 200 MB
- **Screen Resolution:** 800x600 or higher
- **Active Internet Connection**, for license validation and threat signature updates
- **USB 2.0 Port** (optional, depending on deployment method)

Using Breach Remediation

Breach Remediation is designed specifically for use by IT staff. It may be deployed to an endpoint by local insertion of a USB drive which contains the program, or by pushing the program out to the endpoint using psexec, PowerShell, or any other deployment method which you may currently use.

Breach Remediation command line functionality is discussed in detail beginning on page 13.

License Key Status

Breach Remediation uses a license key, which was provided to you upon your purchase of the client. Once registered, the license key is considered active for 14 calendar days – unless a different time interval was specified at time of purchase. Each time the client is used on an endpoint, license status is checked. If your license deactivates (times out), you cannot perform critical operations that the client is intended for. If this occurs, you must re-register the client (see page 14 for further details). This is to prevent unauthorized use. There is no additional cost to re-register the client.

Getting Started

Breach Remediation is shipped as a single executable file (`mbr.exe`). When this executable is run for the first time on the endpoint, it will extract all program and configuration components. Once all components have been extracted, the program is available for use on the endpoint.

For extractions that require 7z dialog suppression, a command line switch (`-gm2`) with the MBBR sfx will suppress the 7z dialog. Below is a sample usage:

```
MBBR-4.3.17.202.exe -gm2
```

MBBR is checking the license key against the Brix licensing server by default. This is for license keys provided to ThreatDown customers. If you are using a license key from Malwarebytes, use the following command to change the licensing server prior to attempting to register your license key. For more information, see [Settings](#).

```
mbr.exe settings -licenseServer:keystone
```

Using an endpoint with a live Internet connection, access a Windows command line prompt and issue the following commands:

```
mbr.exe register -key:<prodkey>
```

You must substitute your license key for `<prodkey>` in the above example. Also, some deployment utilities (e.g., *psexec*) do not support color display as is shown below. When using a utility that does not support color, program messages are displayed in standard monochrome video. Please refer to page 17 for explicit settings regarding color output.

```
Microsoft Visual Studio Debug Console
Malwarebytes Breach Remediation Version: 4.3.9.139
(c) 2022 Malwarebytes. All rights reserved.

Registering product key...

Product key:      ..N9APK
Installation token: ZmeCp745HPfgtoLzcy7j1662764408
Machine id:      70cb31c8546ce8e88656a4f5492d423c3e05380a
Entitlement status: active
Entitlement features:
  feature_set:    default
  key_ttl:        366
  db_ttl:         48
  enable_telemetry: true
Term end date:    2023-10-31T23:59:59.000+00:00
Term type:        subscription
volume_used:      739069
volume_purchased: 1500000

Product key registered successfully
Success
All done!
```

Once the program has been activated, it is necessary to load threat signatures from servers into the client. This enables *Breach Remediation* to detect threats using the most current reference material available. Malware threat signatures are updated several times daily, and rootkit signatures are updated as required.

mbbr.exe update

Database update milestones will be displayed on the screen as the update takes place. When complete, you will see a final status message as well as the database version. That is shown below.

```
Microsoft Visual Studio Debug Console
Malwarebytes Breach Remediation Version: 4.3.9.139
(c) 2022 Malwarebytes. All rights reserved.

Starting update...

Checking for updates...
Found newer version.
Downloading database...
(100% complete)
Applying downloaded updates...
(100% complete)

Current dbVersion 2022.01.26.20 is installed.
Current dbClsPkgVersion 1.0.50327 is installed.

Updated to version 1.0.50327
Latest updates were installed
Success
All done!
```

Once threat signatures have been updated in your local installation, you can use *Breach Remediation* to detect and remove malware from your endpoint. Following is a screenshot of a scan in process.


```
Microsoft Visual Studio Debug Console
Malwarebytes Breach Remediation Version: 4.3.9.139
(c) 2022 Malwarebytes. All rights reserved.
Starting custom scan [ESC=Cancel], [SPACE=Pause/Resume]...

Current Phase:          Scan Complete
Phase Progress:         9/9
Currently Scanning:
Objects Scanned:        35
Malicious Item Detected: Trojan.MBAMTest
Malicious Item Detections: 9
Non-malware Item Detected: PUP.Optional.DotPitch
Non-malware Item Detections: 9
Scan Completion:        100 %
Time Elapsed:           00:00:01:445

Scan completed
Success
All done!
```

While a scan is in process, this screen is constantly updated. Please note the line titled **Phase Progress**. There are nine (9) phases of a scan, which are:

- | | | | |
|---|---------------------|---|--------------------|
| 1 | Pre-scan Operations | 6 | Filesystem Objects |
| 2 | System Drivers | 7 | Custom objects |
| 3 | Memory Objects | 8 | Heuristic Analysis |
| 4 | Startup Objects | 9 | Scan Complete |
| 5 | Registry Objects | | |

This line will reflect each as they are in process. Each phase of the scan requires a different amount of time to complete, so this cannot be used as a method of estimating how long a scan will take to complete.

Interactions with Anti-Rootkit Scanning

When called upon to perform anti-rootkit scanning, *Breach Remediation* uses a special driver which may be incompatible with driver versions used by the anti-malware client and/or *Anti-Rootkit*. If this occurs, *Breach Remediation* must unload the incompatible driver so that it may load its own version. The only way the driver can be gracefully unloaded is by terminating the program which loaded the driver.

This will only occur during active scans by *Anti-Rootkit* or by *Anti-Malware* version 2.0 and above (free, trial or premium), or *Anti-Malware* version 2.0 and above using real-time protection (trial or premium).

If *Anti-Malware* was terminated to allow *Breach Remediation* to run **and** a reboot was required to remove threats detected by *Breach Remediation*, protection will return to its normal state after the reboot. If a reboot is not required, you must manually restart *Anti-Malware* to regain the real-time protection that was turned off temporarily.

Remediation Now or Later?

Breach Remediation offers two types of scans which may be executed. It also offers the capability to automatically decide which threats should be removed, or to allow the user to override default remediation methods selected by the program. This may be valuable in many circumstances, including:

- General assessment of an endpoint’s health with regard to malware
- Ability to collect and analyze evidence of infections
- Exclusion of known false positives

Scans may be executed for the purpose of remediation, or for diagnostic discovery. A remediation scan combines a scan with a remediation method, so that detected threats may be immediately cleaned from the endpoint. A diagnostic scan omits the remediation method, so that a scan is executed, and results are reported. The user may

then determine how to proceed. This may be valuable if you wish to assess the general health of an endpoint, or if you wish to collect data about one or more endpoints without eliminating evidence that you may wish to retain.

These capabilities are listed below.

Remediation Scan

A *remediation scan* combines a scan with an automatic remediation method, so that detected threats may be immediately cleaned from the endpoint. No user intervention is required once the scan begins. This method requires the **–remove** parameter to be specified in order to perform remediation as expected.

Diagnostic Scan

A diagnostic scan is executed when the **–remove** parameter is not used as part of the **scan** command. Scan results are saved automatically to intermediate file **ScanResults.json** in the **./ScanResults** Subdirectory in the directory that *mbr* was executed from. This file is saved in JSON format. Using an JSON processor or editor of your choice, you may inspect scan results at any time until the next scan is executed.

Selective Remediation

If you determine that removing threats detected from the last executed scan should be performed, modify **ScanResults.json** to suit your needs, and run another scan using the **–removelastscan** parameter to remove detections.

As mentioned in the previous section, the scan saved “working data” pertaining to any detected threats in an intermediate file named **ScanResults.json**, in the **./ScanResults** Subdirectory in the directory that *Breach Remediation* was executed from. The user may open and inspect this file (using a JSON processor or editor of their choice) to determine if any detected threats should not be remediated. The **cleanAction** for such traces can be set to **ignore** to exclude them from remediation.

This **ScanResults.json** file will remain after processing has been completed but will be overwritten the next time that *Breach Remediation* executes a scan.

Following is a sample **ScanResults.json** file generated by the scan. Indentation has been added to the JSON file to improve readability.

```
3AB773695FCBB17FA512D6D6F3EFEC654A586B4423C8CC737981A454E32B5BF2
{
  "applicationVersion" : "4.3.9.139",
  "chromeSyncResetQueryRequested" : false,
  "chromeSyncResetQueryResult" : false,
  "clientId" : "mbr",
  "clientType" : "other",
  "componentsUpdatePackageVersion" : "1.0.0",
  "coreDllFileVersion" : "3.0.0.1098",
  "cpu" : "x64",
  "dbSDKUpdatePackageVersion" : "1.0.36317",
  "detectionDateTime" : "2021-09-08T20:21:04Z",
  "fileSystem" : "NTFS",
  "id" : "4acba3e2-10e2-11ec-b40d-00e04c680014",
  "isUserAdmin" : true,
  "licenseState" : "licensed",
  "linkagePhaseComplete" : true,
  "loggedInUserName" : "CLW-3CCNLH2\\jbuonasera",
  "machineID" : "",
  "os" : "Windows 10 (Build 19042.1165)",
  "schemaVersion" : 19,
  "sourceDetails" : {
    "aggressiveMode" : false,
    "clientMetadata" : {
      "jobId" : "",
      "scheduleId" : "",
      "scheduleTag" : ""
    }
  },
}
```

```

"ddsigEnabled" : true,
"filesScannedByIG" : 0,
"objectsScanned" : 1,
"scanEndTime" : "2021-09-08T20:21:05Z",
"scanOnlineStatus" : "unknown",
"scanOptions" : {
  "pumHandling" : "detect",
  "pupHandling" : "detect",
  "scanArchives" : true,
  "scanFileSystem" : true,
  "scanMemoryObjects" : false,
  "scanPUMs" : true,
  "scanPUPs" : true,
  "scanRookits" : false,
  "scanStartupAndRegistry" : false,
  "scanType" : "custom",
  "useHeuristics" : true
},
"scanResult" : "completed",
"scanStartTime" : "2021-09-08T20:21:04Z",
"scanState" : "completed",
"shurikenEnabled" : true,
"type" : "scan"
},
"threats" : [
  {
    "ddsigFileVersion" : "",
    "linkedTraces" : [
      ],
    "mainTrace" : {
      "archiveMember" : "test-trojan.exe",
      "archiveMemberMD5" : "3B9269B0C31CA2CCFB30D75A83B0609E",
      "cleanAction" : "quarantine",
      "cleanContext" : {
        },
      "cleanResult" : "notStarted",
      "cleanResultErrorCode" : 0,
      "cleanTime" : "",
      "generatedByPostCleanupAction" : false,
      "hubbleRequestErrorCode" : 0,
      "id" : "4b1599de-10e2-11ec-8cdf-00e04c680014",
      "igExitCode" : "",
      "isPEFile" : true,
      "isPEFileValid" : true,
      "isWhitelistedByAdsInfo" : false,
      "linkType" : "none",
      "objectMD5" : "556C022F4079280CB526D64A13D576D6",
      "objectPath" : "C:\\TEMP\\TEST_EXAMPLE\\TEST-TROJAN.ZIP",
      "objectSha256" : "D1A99589E25F4382EBC213FD441436A725362F485BD41A6363DC812619B8DC07",
      "objectSize" : -1,
      "objectType" : "file",
      "resolvedPath" : "",
      "suggestedAction" : {
        "archiveDir" : false,
        "chromeExtensionOther" : false,
        "chromeExtensionPreferences" : false,
        "chromeExtensionSecurePreferences" : false,
        "chromeExtensionSyncData" : false,
        "chromeUrlOther" : false,
        "chromeUrlSecurePreferences" : false,
        "chromeUrlSyncData" : false,
        "chromeUrlWebData" : false,
        "disableHubbleWhiteListing" : false,
        "disableSignatureWhiteListing" : false,
        "fileDelete" : true,
        "fileReplace" : false,
        "fileTxtReplace" : false,
        "folderDelete" : false,
        "isChromeObject" : false,
        "isDDS" : false,
      }
    }
  }
]

```

```

        "isDoppleganging" : false,
        "isExternalDetection" : false,
        "isPUP" : false,
        "isShuriken" : false,
        "isWMIEventConsumer" : false,
        "killProcess" : false,
        "minimalWhiteListing" : false,
        "moduleUnload" : false,
        "noLinking" : true,
        "physicalSectorReplace" : false,
        "priorityHigh" : false,
        "priorityNormal" : false,
        "priorityUrgent" : false,
        "processUnload" : false,
        "regKeyDelete" : false,
        "regValueDelete" : false,
        "regValueReplace" : false,
        "shortcutReplace" : false,
        "silentMode" : false,
        "singleDelete" : false,
        "treatAsRootkit" : false,
        "useDDA" : false,
        "verifyResolvedPath" : false,
        "whitelistCheckError" : false
    },
    "winVerifyTrustResult" : {
        "expectedError" : false,
        "lastErrorCode" : 0,
        "wvtCalled" : true,
        "wvtResult" : 0
    }
},
"ruleID" : 145300,
"ruleString" : "",
"rulesVersion" : "1.0.36317",
"srcEngineComponent" : "ame",
"srcEngineThreatNames" : [

],
"threatID" : 3349,
"threatName" : "Trojan.MBAMTest"
}
],
"threatsDetected" : 1
}

```

Please note that each detected threat is documented in the same manner, which allows for easy identification of data within the file. A JSON tag titled `<cleanAction>` contains a remediation method and is preset to the value **quarantine**. You may not wish to quarantine the detected threat. Here are the three choices which are available to you.

- **remove:** Delete the malware in its present form, render it harmless, and move it to Quarantine
- **delete:** Delete the malware from the endpoint without copying it to Quarantine
- **ignore:** Leave the file intact in its current location.

Once you are done inspecting/editing this file, you may resume or cancel the selective remediation scan. If you cancel the remediation phase of the scan, the intermediate file will be retained. If you continue the remediation phase, the intermediate file will be used to control remediation, and will be deleted once remediation is complete.

In the screenshot below, command `mbsr scan -path:C:\temp\TestFiles_32 -remove` was executed to provide a listing of the files removed and placed into Quarantine.

```
Administrator: C:\Windows\system32\cmd.exe
Malwarebytes Breach Remediation Version: 4.3.9.139
(c) 2022 Malwarebytes. All rights reserved.

Starting custom scan [ESC=Cancel], [SPACE=Pause/Resume]...

Current Phase:          Scan Complete
Phase Progress:         9/9
Currently Scanning:
Objects Scanned:        13
Malicious Item Detected: Malware.Heuristic.1001
Malicious Item Detections: 3
Non-malware Item Detected: PUP.Optional.DotPitch
Non-malware Item Detections: 3
Scan Completion:        100 %
Time Elapsed:           00:00:00:251

Items Removed: 6
Scan completed
Success
All done!
```

Scan Summary

After every scan, MBBR also creates a ScanSummary.txt file along with ScanResults.json. The ScanSummary file gives a high-level overview about the scan results including scan type – (Threat, Hyper, Full), scan start time in UTC format, number of objects scanned, number of items detected and remediated, and the total elapsed scan time. Every new scan would overwrite the previous ScanSummary.txt file.

Following is a sample ScanSummary.txt file generated after a **Diagnostic only** scan:

```
ScanType: Threat Scan
Scan Time: 2021-05-15T15:04:44Z
Scan Result: Completed

Objects Scanned: 262396

Malicious items detected: 0

Non-malicious items detected: 0

Time Elapsed: 45 sec
```

Following is a sample ScanSummary.txt file generated after a **Remediation** scan:

```
ScanType: Threat Scan
Scan Time: 2021-05-15T15:45:02Z
Scan Result: Completed

Objects Scanned: 262478

Malicious items detected: 2
    Heuristics.Shuriken, C:\USERS\NIK\DESKTOP\TEST.ORACLE.EXE
    Trojan.MBAMTest, C:\USERS\NIK\DESKTOP\TEST-TROJAN.EXE
```

```
Non-malicious items detected: 1
  PUP.Optional.DotPitch, C:\USERS\NIK\DESKTOP\TEST_PUP.EXE

Items removed: 3
  C:\USERS\NIK\DESKTOP\TEST_PUP.EXE
  C:\USERS\NIK\DESKTOP\TEST.ORACLE.EXE
  C:\USERS\NIK\DESKTOP\TEST-TROJAN.EXE

Time Elapsed: 1 min, 0 sec
```

Excluding Items from Scanning

It is not uncommon to have legitimate items stored on your endpoints which may be identified as threats by anti-virus or anti-malware software. *Breach Remediation* recognizes that and offers two methods to exclude those items from scanning. Those methods are:

- **Exclude by specification** – This method allows interactive or scripted exclusion of files, folders, and wildcards.
- **Exclude List** – In addition to the previous method, this method allows exclusion of file extensions, registry keys, registry values, and vendor (the name which is used to identify threats). Items to be excluded are enclosed in one or more JSON files. A sample exclusion list is shown here. Please note that indentation has been added here to aid in understanding. The JSON files can be specified as either local file paths or web URLs.

Following is a sample **exclusions.json** file generated by the scan. Indentation has been added to the JSON file to improve readability.

```
{
  "exclusions" : [
    {
      "path" : "dll",
      "type" : "extension"
    },
    {
      "path" : "c:\\myprog\\test.exe",
      "type" : "file"
    },
    {
      "path" : "c:\\myprog\\abc",
      "type" : "folder"
    },
    {
      "path" : "HKLM\\SYSTEM\\CURRENTCONTROLSET\\SERVICES\\1394843d",
      "type" : "regkey"
    },
    {
      "path" : "HKCU\\SOFTWARE\\MICROSOFT\\WINDOWS\\CURRENTVERSION\\RUN|DESKBAR",
      "type" : "regval"
    },
    {
      "path" : "**myprog*",
      "type" : "wildcard"
    }
  ],
  "path" : "Trojan.MBAMTest",
  "type" : "vendor"
},
"schemaVersion" : 1
}
```

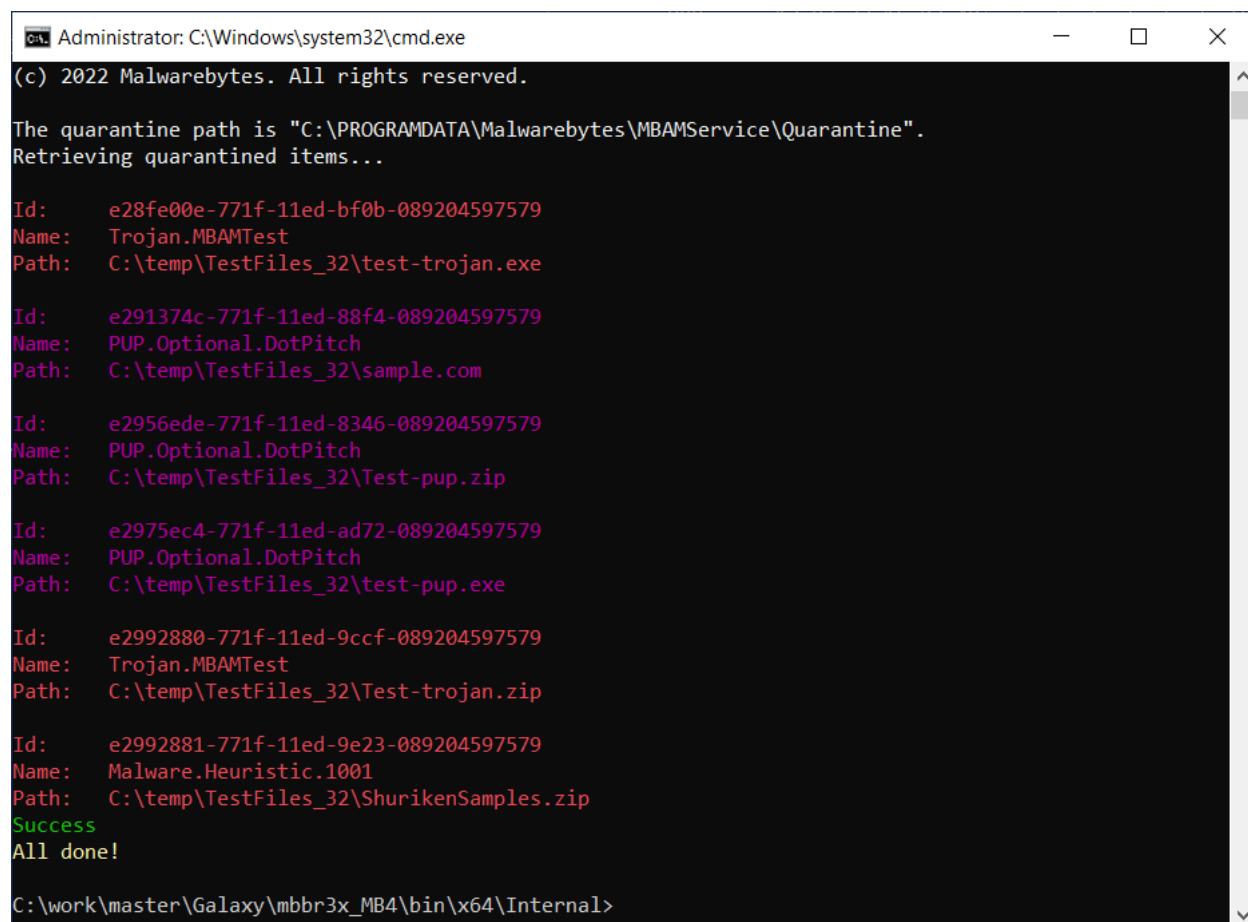
Please see pages 15-16 for complete details.

Restoring Items from Quarantine

Breach Remediation offers several different methods of restoring items from Quarantine. You may choose any of the following methods:

- **Restore all** – Restores all items currently stored in Quarantine to their original locations
- **Restore by id** – This method utilizes a screen/file-based list to selectively restore items to their original locations. This is typically a manual operation, though it may also be performed using a script. The list of items changes after each restore operation, so the list must be recreated when multiple restore operations are required.
- **Automated restore** – The JSON list of items stored in Quarantine is interactively modified to specify which items will be selectively restored. The modified list is then used programmatically to perform the restore operation.

In the screenshot below, command `mbr quarantine -list` was executed to provide a listing of the contents of Quarantine. This list shows the index number that can be used for restoration, the name of the threat that was placed into Quarantine, and its original location before it was placed in Quarantine.



```
Administrator: C:\Windows\system32\cmd.exe
(c) 2022 Malwarebytes. All rights reserved.

The quarantine path is "C:\PROGRAMDATA\Malwarebytes\MBAMService\Quarantine".
Retrieving quarantined items...

Id:      e28fe00e-771f-11ed-bf0b-089204597579
Name:    Trojan.MBAMTest
Path:    C:\temp\TestFiles_32\test-trojan.exe

Id:      e291374c-771f-11ed-88f4-089204597579
Name:    PUP.Optional.DotPitch
Path:    C:\temp\TestFiles_32\sample.com

Id:      e2956ede-771f-11ed-8346-089204597579
Name:    PUP.Optional.DotPitch
Path:    C:\temp\TestFiles_32\Test-pup.zip

Id:      e2975ec4-771f-11ed-ad72-089204597579
Name:    PUP.Optional.DotPitch
Path:    C:\temp\TestFiles_32\test-pup.exe

Id:      e2992880-771f-11ed-9ccf-089204597579
Name:    Trojan.MBAMTest
Path:    C:\temp\TestFiles_32\Test-trojan.zip

Id:      e2992881-771f-11ed-9e23-089204597579
Name:    Malware.Heuristic.1001
Path:    C:\temp\TestFiles_32\ShurikenSamples.zip
Success
All done!

C:\work\master\Galaxy\mbr3x_MB4\bin\x64\Internal>
```

The following snippet shows partial sample contents of file **RestoreList.json**, which is generated at the same time the screen-based list is created. For demonstration purposes, the file has been edited so that only the first quarantined item will be restored using the JSON file as input to command `mbr.exe quarantine -restorelist`.


```

{
  "quarantinedItems": [
    {
      "action": "none",
      "id": "1a9df7c0-c707-11ec-b659-d46a6a8fddb4",
      "path": "C:\\temp\\TestFiles_32\\test-trojan.exe",
      "threatName": "Trojan.MBAMTest"
    },
    {
      "action": "none",
      "id": "225a58d2-c707-11ec-a353-d46a6a8fddb4",
      "path": "C:\\temp\\TestFiles_32\\Test-trojan.zip",
      "threatName": "Trojan.MBAMTest"
    },
    {
      "action": "none",
      "id": "225cef34-c707-11ec-9230-d46a6a8fddb4",
      "path": "C:\\temp\\TestFiles_32\\sample.com",
      "threatName": "PUP.Optional.DotPitch"
    },
    {
      "action": "none",
      "id": "226269aa-c707-11ec-bd91-d46a6a8fddb4",
      "path": "C:\\temp\\TestFiles_32\\Test-pup.zip",
      "threatName": "PUP.Optional.DotPitch"
    },
    {
      "action": "none",
      "id": "2264fff8-c707-11ec-833c-d46a6a8fddb4",
      "path": "C:\\temp\\TestFiles_32\\test-pup.exe",
      "threatName": "PUP.Optional.DotPitch"
    },
    {
      "action": "none",
      "id": "227756c6-c707-11ec-a0de-d46a6a8fddb4",
      "path": "C:\\temp\\TestFiles_32\\ShurikenSamples.zip",
      "threatName": "Malware.AI.2928521561"
    }
  ],
  "schemaVersion": 1
}

```

The above is merely an introduction to the methods. Please see page 18 for more detailed usage guidance.

Command Line Parameters

Breach Remediation supports a variety of command line parameters, which can be used from a command prompt, batch file or script. When used from a script, additional commands may be required to support the scripting model being used.

Conventions

The command line structure uses modifiers. These are shown as hyphens (-) immediately preceding parameters. Multiple modifiers may be combined with a parameter. When multiple parameters are used, they must be separated by spaces. In addition, the following conventions are used:

- **text without brackets or braces**
Items you must type as shown
- **<text inside angle brackets>**
Required information for which you must supply a value
Example: **mbr.exe <parameter_1>**
- **[text inside square brackets]**
Optional items
Example: **mbr.exe [parameter_1]**
- **Grouping of dots (...)**
A set of specifications
Example: **mbr.exe <parameter_1> [parameter_2] ... [parameter_n]**
- **{text inside braces}**
A set of required items; choose one from the list provided
Example: **mbr.exe {0 | 1 | 2 | 3}**
- **vertical bar (|)**
Separator between mutually exclusive items; choose one
Example: **mbr.exe <0 | 1 | 2 | 3>**

Command Line Overview

Breach Remediation commands are specified in the following format:

mbr.exe { register | update | version | scan | errorout | quarantine | settings } [options]

Following is a list of high-level commands which may be executed. Each command is detailed beginning on the following page.

register	Using your license key, this unlocks the features of <i>Breach Remediation</i> . This will also show license status.
update	Updates the client's threat signature databases.
version	Displays the program version number.
scan	Scans the endpoint for malware and optionally removes malware found during the scan.
errorout	Specifies where error output is directed to.
quarantine	Controls program actions related to threat quarantine activities.
settings	Used to specify universal program settings. These settings are persistent and are used for color and proxy functionality.

In addition, you may type **mbr** without any additional specifications to see a list of valid commands. This list will span multiple windows if the command line is launched to its default size, so you will achieve best results by stretching the window to show more command line dialog at one time.

Command Line Reference

Commands listed here are listed individually. Each command performs tasks according to parameters. These are primarily used by a system administrator via script, batch file, GPO update, or remote desktop. The admin may configure *Breach Remediation* to operate as a remote task, invisible to the endpoint user.

register

Usage:

mbr.exe register [-key:<prodkey>]

Purpose:

Specifies the unique license key assigned to the partner or customer. This will be passed to the licensing server for validation to ensure it is active (non-expired). If the key is valid and the license is active, it will also display status about the license, such as expiration date, volume purchased, volume used, etc.

If the key is active, the local installation will operate with this status for 14 calendar days (or the time interval specified in your license agreement). This “Last Known Good” status is persisted on the USB or wherever the binaries are stored. This allows the USB installation to work as if it were fully registered on offline endpoints or without needing the key. **A live Internet connection is required.**

If **-key** is not specified, license status and the expiration date/time will be displayed. **Please note** that if the key is not active, the user may not update threat signature databases, scan for malware, list contents of quarantine, or restore files from quarantine.

Parameters:

-key:<prodkey>

Specification of <prodkey>, the license key assigned to the user.

update

Usage:

mbr.exe update

Purpose:

Updates local threat signature databases. This command will result in an error condition if (a) the license is not active, or (b) if no active Internet connection is available. If threat signature databases have expired (timed out), this command must precede execution of a scan. If a proxy server is needed to access the Internet, you must run the **proxy** command before attempting to perform the update.

Parameters:

none

version

Usage:

mbr.exe version

Purpose:

Displays the version number of the *Breach Remediation* installation.

Parameters:

none

scan

Usage:

```
mbbr.exe scan [-full | -threat | -hyper | -path:<path>]
              [-exclude:<paths>]
              [-excludelist:<exclists1>;<exclistsN>]
              [-excludelist:<https://url.com/exclists >]
              [-noarchive]
              [-ignorepu]
              [-stopondetect [-malware:<cnt>] [-pu:<cnt>]]
              [-tag:<tagname>]
              [-stdlog:<filepath>]
              [-noscanresults]
              [-pfi:<secs>]
              [-useExpertSystemsAlgo]
              [-ark]
              [-remove [-noreboot]
              [-removelastscan [-noreboot]
              [-stdout:{off | detail | summary}]
              [-lowimpact]
```

Purpose:

Executes a scan based on parameters specified. If the program license is inactive, attempts to perform a scan will result in an error. Current threat signature databases are also required. If this command is executed without a directive to quarantine detected threats, scan results are saved to file **ScanResults.JSON**. This file is found in the **./ScanResults** Subdirectory in the directory that *Breach Remediation* was executed from.

Parameters:

-full

A full scan includes all scanning capability which *Breach Remediation* has to offer.

-threat

This will examine “hotspots” on an endpoint for malware without analyzing the entire endpoint. If **-remove** is specified, any threats found during the scan will be quarantined.

-hyper

Initiates a scan focusing only on Memory Objects and Heuristics to determine if malware is actively running on the endpoint. If **-remove** is specified, any threats found during the scan will be quarantined.

-path:<path>

Semicolon-delimited list of folder paths to scan on the endpoint’s file system. Use double-quotes for paths that contain spaces. Paths specified are recursive, so subfolders will also be included automatically. The presence of the paths specified will be verified, and an error will result if the paths do not exist. Similarly, an error will be generated if the paths are encrypted, as the program is not capable of decrypting the path. If **-remove** is specified, any threats found during the scan will be quarantined. **Please note** that this method scans only the folder path(s) which have been specified. This scan type **does not** include memory, startup modules or heuristic analysis.

-exclude:<paths>

Excludes specific files, folders, and wildcard specifications from scanning. **Please note** that the following rules apply:

- Multiple exclusion items must be separated by semicolons.
- Wildcards may be used for files and folders using existing Windows wildcard standards.

- Wildcards may also be used **within** registry keys to add flexibility. Please see this style in the example shown on page 9.
- Files specified without a directory path (e.g., file.exe) must be in the directory where *Breach Remediation* is installed.
- Files and/or folders which contain embedded spaces must be enclosed in double quotes. As an example, references to `C:\Program Files\file.exe` should instead be specified as `"C:\Program Files\file.exe"`.

`-excludelist:< exlists1 >;< exlistsN >` – Use exclusion list(s) stored locally on the machine where mbbr is executing.

`-excludelist:<https://url.com/exclists >` – Use to pull down exclusion list from a URL.

Provides for exclusions itemized within one or more JSON files specified as parameters for this command. Files have no naming convention (filename or extension), but they must be in JSON format. The following items may be excluded using exclusion lists:

- Files
- Extensions
- Threat Vendor
- Folders
- Registry Keys
- Wildcards
- Registry Values

Please note that the following rules apply:

- Multiple items (within the `<Path>` element) must be separated by semicolons.
- Files specified without a directory path (e.g., file.exe) must be in the directory where *Breach Remediation* is installed.
- When excluding a registry value (*regval*), it must be preceded by the registry key (*regkey*), delimited by the pipe “|” character.
- Extensions pertain to the entire file system that is subject to scanning.
- Threat vendor is specific to our definitions.
- Multiple exclusion list files must be separated by semicolons.
- Wildcards may be used for files and folders using existing Windows wildcard standards.
- Wildcards may also be used **within** registry keys to add flexibility. Please see this style in the example shown on page 9.

A sample Exclusion List is shown on page 9.

`-noarchive`

By default, the contents of archives (zip, rar, etc.) are scanned. Use this option to disable archive scanning. *Breach Remediation* will stop scanning an archive if it finds a single infected file and will quarantine the entire archive file.

`-ignorepu`

Instructs the scanner to ignore all Potentially Unwanted Programs (PUPs) and Potentially Unwanted Modifications (PUMs) that may be installed on the target endpoint.

`-stopondetect [-malware:<cnt>] [-pu:<cnt>]`

Instructs the scanner to terminate the scan when a certain number of either malware or potentially unwanted items are found. This allows a quick determination of whether the endpoint is infected without requiring a full scan to be performed. If `<cnt>` is not reached, the scan will run to completion.

Specify counts for either malware threats, PUP/PUM or both. The scanner will terminate when either criterion is met.

If `-stopondetect` is specified, at least one of the `-malware` or `-pu` options must also be specified.

`-tag:<tagname>`

This text string will be sent along with all usage data to our billing system. If the string includes embedded spaces, it must be surrounded by double quotes (“”). It will help you to associated billing events with your billing system. Typical usage would be for you to add the Job ID or Store ID or Employee ID or all of these, so that you can see these on your invoice.

–stdlog:<filepath>

Specifies the log location for normal output. If not specified, normal output will be written to `.\logs\MBBR-STDOUT.JSON`. Use double-quotes (“”) for paths that contain spaces. **Please note** that use of the default specification for <filepath> will result in loss of scan data for any scans previously executed, as a new scan overwrites existing results.

–noscanresults

Instructs the scanner to disregard creation of intermediate file `ScanResults.json`. This parameter disables the ability to selectively remediate items during a scan and is only valid for diagnostic scans.

–pfi:[<secs>]

Controls the frequency at which log file `ScanProgress.json` is updated. This log file is stored in the folder specified by `–stdlog`. File creation frequency is in the range of 1-3600 seconds. This file is only created when `–stdout` is set to summary. If this option is not specified, no Scan Progress file will be created.

–useExpertSystemsAlgo

When a scan is run with this switch, it enables aggressive detection technology based on AI-expert systems algorithms.

–ark

Enables Anti-rootkit scanner functionality to be used during the scan. Any rootkits found will be removed if `–remove` is specified.

–remove

Instructs the scanner to quarantine malware, PUPs and PUMs found during the scan. This parameter is not allowed if `–stopondetect` is specified. If `–remove` and `–noreboot` are both specified in a scan command and the scan detected threats during execution, a warning message will be displayed after the scan has completed to notify the user that a reboot is required to remove the threat(s) from the endpoint.

–noreboot

Some malware executes in a manner that requires a reboot to complete the removal process. If this occurs, the scanner will automatically reboot the system (after a delay specified by the `scan.rebootwait` parameter associated with the settings command). If an immediate reboot is not desired, use this option. Please note that certain malware may not be fully removed if this option is used. If `–remove` and `–noreboot` are both specified in a scan command and the scan detected threats during its execution, a warning message will be displayed after the scan has completed to notify the user that a reboot is required to remove the threat(s) from the endpoint.

–removelastscan

This parameter is associated with a *diagnostic scan*. Instead of executing a new scan, it instructs the scanner to use intermediate file `ScanResults.xml` to remediate threats detected during the last scan that was executed.

–stdout:{off | detail | summary}

Controls the level of output to the console. Defaults to **summary** if not specified.

–lowimpact

Low impact scans run at a lower system priority, minimizing the impact on the foreground system usage. This allows the user to perform other tasks on the system normally without any

noticeable impact from the scan. A scan with this option may take a little longer to complete than a scan without this option.

errorout

Usage:

```
mbr.exe errorout [[-console:{on | off}] [-delete] [-errlog:<file>] | [-reset]]
```

Purpose:

Specifies where error output will be directed to. Values set using this command will persist until they are cleared or modified. Issuing this command without arguments will display current settings.

Parameters:

`-console:{off | on}`

Specifies if error output is displayed on the console. Default value is ON.

`-delete`

Deletes the output file, if it exists. This command uses the default error log location, unless the error log location has been changed using the `-errlog` switch.

`-errorlog:<file>`

Specifies the log location for error output. This will overwrite any previously specified location. If `<file>` contains any embedded spaces, please enclose `<file>` in double quotes ("). The default location is `.\logs\MBBR-ERROUT.TXT`.

`-reset`

Reverts settings associated with this command back to default values.

quarantine

Usage:

```
mbr.exe quarantine [-list]
                    [-path:<path>]
                    [-resetpath]
                    [-restoreall]
                    [-restore:<itemIDs>]
                    [-restorelist]
```

Purpose:

Set/reset location of quarantine, list quarantine contents, and restore files from quarantine. Use this command without any additional arguments to display the current quarantine location.

Parameters:

`-path:<path>`

Specifies where quarantined content will be stored after this command has been executed. This replaces any previously specified location. If `<path>` contains embedded spaces, enclose `<path>` in double quotes ("). **Please note** that content quarantined prior to execution of this command will not be moved.

`-resetpath`

Causes the quarantine file folder to revert to the default folder. Files stored in quarantine prior to execution of this command will not be moved to the default folder. The default quarantine folder is:

```
C:\ProgramData\Malwarebytes\MBAMService\Quarantine
```

–list

Shows the current quarantine location, lists contents of the quarantine to screen output, and generates file **RestoreList.json** for use by the **–restorelist** option. In addition, a unique **ID** is associated with each item which has been moved to quarantine. The ID is used primarily in conjunction with the **–restore:<itemIDs>** option, to simplify manual restore operations.

–restoreall

Restores all quarantined items to their original locations.

–restore:<itemID1> [,itemID2] ... [,itemIDn]

Restores one or more items from the list of quarantined items shown on the screen (or in file **RestoreList.json**). Items are specified by their ID. When multiple items are to be restored via a single execution of this command, their IDs should be separated by commas without delimiting spaces. **Please note** that execution of this command will delete file **RestoreList.json**, and that any pre-existing item IDs still shown on the screen are no longer valid. You must exercise the **–list** option again prior to any subsequent execution of this option.

–restorelist

Using file **RestoreList.json** as a guide, this command restores specified items to their original location from Quarantine. For each item to be restored, modify the **<Action>** element associated with the item to be restored, changing the value to **restore**. Unless changed by the user (or by a third-party application), the default value of the **<Action>** element is **none**. A sample **RestoreList.json** file is shown on page 12.

settings

Usage:

```
mbr.exe settings [-color:off|on]
                [-scan.rebootwait:<seconds>]
                [-scan.rebootmsg:<message>]
                [-proxy.clear]
                [-proxy.enabled:true | false]
                [-proxy.server:<host>]
                [-proxy.port:<port>]
                [-proxy.user:<user>]
                [-proxy.password:<password>]
                [-log.enabled:true|false]
                [-log.server:<host>]
                [-log.port:<port>]
                [-log.netprotocol:<tcp|udp>]
                [-log.allevents:on|off]
                [-log.events:<eventname>:on|off]
                [-log.events:<eventname>:on|off]
                [-log.test]
                [-log.clear]
                [-customdb.enabled:true | false]
                [-customdb.add:<name>:<MD5 Hash>]
                [-customdb.load:<openIOCFiles>]
                [-customdb.clear]
                [-customdb.list]
                [-resetAll]
                [-useStaticURLs:<true|false>]
                [-scan.backupResults:<num>]
                [-forceRun:<true/false>]
                [-licenseServer:<keystone|brix>]
```

Purpose:

Used to define several program settings that will be universally used by *Breach Remediation*. Execution of this command with no options specified will display current settings. If no modifications have been made, default values will be shown. Please note that changes to settings require administrative privileges. All users may inspect current settings.

Parameters:

`-color:off|on`

Specifies whether program output can utilize color, or if display will be limited to monochrome. Applications which attempt to capture standard output and error output of *Breach Remediation* may encounter issues. Turning color off solves this problem.

`-scan.rebootwait:<seconds>`

Amount of time to wait before a reboot if a reboot is required to remove threats detected during a scan. The maximum time is 300 seconds, and the default time is 60 seconds.

`-scan.rebootmsg:<message>`

Text message to be displayed prior to a reboot if a reboot is required to removed threats detected during a scan. The maximum message length is 100 characters. If a message contains embedded spaces, the text string must be bounded by double quotes.

`-proxy.clear`

Clears all settings associated with proxy servers.

–proxy.enabled:true | false

Enables or disables use of a proxy server for external Internet access. Internet access is required for program updates as well as threat signature updates. If set to **true**, the proxy **host** and **port** must also be specified. **PLEASE NOTE:** If your network requires use of a proxy server for Internet access, proxy settings must be defined and enabled before a **register** or **update** command may be successfully executed.

–proxy.server:<host>

Hostname and/or IP address of proxy server providing external Internet access.

–proxy.port:<port>

Port number on proxy <host> which is used for external Internet access.

–proxy.user:<user>

Username for proxy usage, if authentication is required.

–proxy.password:<password>

Password for username <user>, if authentication is required to use proxy.

–log.enabled:true|false

Specifies whether program execution will be logged to a syslog server. All data utilizes a CEF (Common Event Format) standard. If this parameter is set to **true**, the syslog *host* IP/FQDN and *port* number must also be specified before event logging can take place.

–log.server:<host>

IP address or Fully Qualified Domain Name (FQDN) of a syslog server which will receive event logs generated by *Breach Remediation*. A valid *port* number must also be specified before logging can take place.

–log.port:<port>

Valid port number for the syslog server which will receive event logs generated by *Breach Remediation*. A valid syslog *host* specification must also be specified before logging can take place.

–log.netprotocol:<tcp|udp>

Specifies whether the TCP or UDP protocol is used to send data to a syslog server. TCP is the default.

–log.allevents:on|off

Sets all syslog events On or Off. All events are on by default.

–log.events:<eventname>:on|off

Specifies whether logging is enabled/disabled for each individual potential event which may be logged by *Breach Remediation*. These events are discussed at length beginning on page 29, and are itemized here:

- ScanStartEvent
- DetectionEvent
- RestoreEvent
- CustomDbUpdateEvent
- ScanEndEvent
- RemovalEvent
- RestoreStartEvent
- DbUpdateEvent
- RemoveLastScanEvent
- RestoreEndEvent

–log.test

Attempts to make contact with the syslog server using the *host* and *port* specifications which have been provided. Results of the connection attempt are shown on-screen. This command does not generate an event log entry. No additional parameters are required.

-log.clear

Clears all Log settings.

-customdb.enabled:true | false

Specifies whether custom database rules are enabled (true) or disabled (false). Default value is false.

-customdb.add:<name>:<MD5 Hash>]

Allows a single MD5 hash to be added to the Rules database without requiring use of the JSON file as an input medium. The MD5 hash specified here is incrementally added to existing rules.

-customdb.load:<JsonFiles>

Loads one or more JSON files into the Rules database. When multiple files are specified, they must be separated by semicolons. If embedded spaces are present in the file and/or path specification, the full path and file must be enclosed by double quotes. OpenIOC must be converted to the required JSON format to be used with this command. A **SampleCustomRules.json** file is provided in the mbr directory for reference.

-customdb.clear

Deletes all existing custom rules.

-customdb.list

Lists all existing custom rules.

-resetAll

Reset all settings to defaults.

-useStaticURLs:<true|false>

Enable/disable use of static URLs for registering and updates. We use both Dynamic and Static DNS addresses for the licensing server used to register applications, and for the update server used to provide updates. Dynamic DNS resolves to IP addresses that are not fixed and are subject to change, while Static DNS resolves to a set of fixed IP addresses. In some customer environments, outgoing connections may be restricted. They must explicitly whitelist these connections, and in some cases, the firewall/network may require the specific IP addresses (not the DNS names) to do so. In such environments, excluding the following static IP addresses and enabling the “-useStaticURLs” option will allow *Breach Remediation* to reach the servers.

List of Static IPs:

34.234.177.227

54.144.41.233

52.22.229.44

54.205.182.14

-scan.backupResults:<num>

Enables mbbbr to create some # of backup of Scan Results on subsequent scans instead of overwriting with a new file and losing the existing file. Default is 0.

`--forceRun:<true|false>`

Determines whether MBBR should run when other products such as our consumer versions are installed and actively running on the machine. If `--forceRun` is set to *true* (default) before starting the scan, a warning message will be logged and the scan will proceed. This is currently the default setting and operation. If `--forceRun` is set to *false* before starting the scan, a more detailed warning message will be logged and displayed to the user—asking the user to exit the App and stop the Service, and then mbbbr application will terminate.

`--licenseServer:<keystone|brix>`

Changes the license server that MBBR checks the entered license key against. Default is brix, which is used for ThreatDown license keys. Try changing the license server to keystone if your license key is not working or is a Malwarebytes license key.

Scan Log

Finalized results of scans executed by *Breach Remediation* are saved in file MBBR-STDOUT.JSON, a scan log which may be imported by several document formats as well as by Internet-based applications. The JSON includes details on the scan and any threats detected during the scan. The root JSON element in the log file is a unique hash of the results file. All subsequent data is grouped by section. Those sections – and data related to those sections – are described here.

header Section

This section provides high-level information about the scan that was performed and is placed at the top of the json.

applicationVersion

Version of Breach Remediation being used for the scan.

clientID

The application client used to run this scan, which would be “mbbr” in this case.

clientType

The type of client used to run this scan. (For internal use only).

componentsUpdatePackageVersion

The version of internal components used for the scan.

cpu

CPU architecture of the system. Valid values are x86 (32-bit) or x64 (64-bit).

dbSDKUpdatePackageVersion

Malware threat signature database version being used for the scan.

detectionDateTime

The time at which this scan log was created. This is in ISO 8601 UTC format.

FORMAT: *yyyy-mm-dd Thh:mm:ssZ*, where:

- yyyy*=Year
- mm*=Month
- dd*=Date
- hh*=Hours
- mm*=Minutes
- ss*=Seconds
- Z*=indicates UTC

fileSystem

The file system of the endpoint’s primary disk drive (meaning the drive on which the operating system is loaded). Valid values are NTFS, FAT or FAT32.

id

The unique ID of this scan.

isUserAdmin

Indicates if the scan was run with admin privileges. Expected to be “true” for all mbbbr scans.

licenseState

Specifies the license state of the client. Valid value for the remediation client is **licensed**.

linkagePhaseComplete

(For internal use only)

loggedInUserName

Windows username associated with execution of the scan.

machineID

This is not used in the remediation client and can be ignored.

OS

Operating System version in use on the endpoint being scanned. This field will also include Service Packs in use (if applicable).

schemaVersion

The version of scan log json schema used *(for internal use only)*.

sourceDetails Section

This section provides summary of the scan as well as information on various categories that were employed during the scan.

aggressiveMode

Indicates if a more aggressive heuristics approach was used during the scan. *(for internal use only)*

clientMetadata

This is not used in the remediation client and can be ignored.

objectsScanned

Number of objects scanned.

scanOnlineStatus

Indicates if system was online at the time of scan. *(for internal use only)*

shurikenEnabled

Indicates if signature-less anomaly detection is enabled. Valid value for remediation client is **true**.

scanState

Indicates if the scan completed successfully or was cancelled.

threatsDetected

Number of threats detected.

scanStartTime

The time at which this scan was started. This is in ISO 8601 UTC format.

FORMAT: *yyyy-mm-dd Thh:mm:ssZ*, where:

yyyy=Year
mm=Month
dd=Date
hh=Hours
mm=Minutes
ss=Seconds
Z=indicates UTC

scanEndTime

The time at which this scan ended. This is in ISO 8601 UTC format.

FORMAT: *yyyy-mm-dd Thh:mm:ssZ*, where:

yyyy=Year
mm=Month
dd=Date
hh=Hours
mm=Minutes
ss=Seconds
Z=indicates UTC

scanResult

Final result of scan. Valid values are **cancelled**, **completed** or **failed**. If a scan was executed with **–stopondetect** and terminated as a result of this specification being set, **scanResult** will be set to **completed**.

scanOptions Section

This section provides information on various categories that were employed during the scan.

pumHandling

Denotes whether scanning of Potentially Unwanted Modifications (PUMs) is active. Set to **enabled** by default. Reverts to **disabled** if **–ignorepu** parameter is specified in scan settings.

pupHandling

Denotes whether scanning of Potentially Unwanted Programs (PUPs) is active. Set to **enabled** by default. Reverts to **disabled** if **–ignorepu** parameter is specified in scan settings.

scanArchives

Denotes whether scanning of archive files is active. This includes ZIP, RAR, ARJ, CAB and 7Z files. Valid values are **enabled** (default value) or **disabled** (when **-noarchive** is specified as part of **scan** command). When enabled, scanning of archiving is limited to three levels deep. When disabled, the archive is scanned as a single file. Encrypted (password-protected) archives cannot be effectively scanned.

scanFilesystem

Denotes whether scanning of the file system is active. Set to **enabled** when **-threat** or **-path** scan types are specified or **disabled** when **-hyper** scan type is specified.

scanMemoryObjects

Denotes whether scanning of running memory processes is active. Set to **enabled** when **-threat** or **-hyper** scan types are specified or **disabled** when **-path** scan type is specified.

scanPUMs

Denotes whether scanning of PUM's (Potentially Unwanted Modifications) is active. Valid values are **Always Detect** (default value), **Warn User** and **Ignore**.

scanPUPs

Denotes whether scanning of PUP's (Potentially Unwanted Programs) is active. Valid values are **Always Detect** (default value), **Warn User** and **Ignore**.

scanRootkits

Denotes whether anti-rootkit scanning is active. Valid values are **enabled** or **disabled** (default value). Value is determined based on setting of scan parameter **-ark**.

scanStartupAndRegistry

Denotes whether scanning of startup-related processes and modules, and registry locations is active. Set to **enabled** when **-threat** or **-hyper** scan types are specified or **disabled** when **-path** scan type is specified.

scanType

Type of scan which was executed. Valid values are **custom**, **threat**, or **hyper**.

useHeuristics

Denotes whether heuristics are employed during scanning. Heuristics enables enhanced detection of threats which may avoid detection by signatures only. This is set to **enabled** when **-threat** or **-hyper** scan types are specified or **disabled** when **-path** scan type is specified.

threats Section

Each threat consists of a *main trace* on which the threat was detected and additional *linked traces* that are linked to the main trace.

There are eight different major categories of traces which may be detected during a scan - file, folder, process, module, regKey, regValue, ADS, physicalSector. Each type has a path and additional detail fields on the trace. Descriptions of all detail fields are as follows. This section provides information on threats which were detected during the scan.

ruleID

ID of the threat signature that detected this threat.

rulesVersion

Malware threat signature database version used for this detection.

threatID

ID of the threat (or threat family), as categorized by the Research Team.

threatName

Name of the threat (or threat family), as categorized by the Research Team. Please note that the same threat may be identified with different names by the various antivirus/antimalware products.

mainTrace Section

There are seven different major categories of traces which may be detected during a scan - file, folder, process, module, regKey, regValue, physicalSector. Each type has a path and additional detail fields on the trace.

Descriptions of all detail fields are as follows. This section provides information on threats which were detected during the scan.

cleanAction

Describes the action taken to remediate the detected trace. Valid values are *quarantine*, *delete* and *ignore*.

cleanContext

Data needed for cleanup.

cleanResult

Result of cleanup. Valid value after a scan is "notStarted."

cleanResultErrorCode

The last error code, if clean result is an error. This will be zero (0) if the cleanup was successful. Any other number indicates an error. The specific error code is useful for debugging.

cleanTime

The time at which cleanup starts.

generatedByPostCleanupAction

(For internal use only)

id

The unique ID of this trace.

linkType

(For internal use only)

objectMD5

The 32-byte MD5 hash of file object, if this is a file trace. For other types of traces, this value is empty.

objectPath

Where the threat was detected. If the threat was found on the Windows filesystem, this contains the full drive/directory/filename. If the threat was found in the Windows registry, this entry contains the registry key/value name/value data corresponding to the threat.

objectSHA256

The SHA256 hash of file object, if this is a file trace. For other types of traces, this value is empty.

objectType

The type of trace – file, folder, process, module, regKey, regValue, physicalSector.

SuggestedAction

Additional data for cleanup.

linkedTraces Section

This section includes any traces linked to the main trace of a threat (e.g., shortcut links to a detected file). The section will be empty if there are no linked traces.

The detail fields of each linked trace are the same as those of the **mainTrace** section.

ruleID

ID of the threat signature that detected this threat.

rulesVersion

Malware threat signature database version used for this detection.

threatID

ID of the threat (or threat family), as categorized by the Research Team.

threatName

Name of the threat (or threat family), as categorized by the Research Team. Please note that the same threat may be identified with different names by the various antivirus/antimalware products.

Sample Scan Progress File

The following is a sample Scan Progress log created during a scan. When requested, these are generated at regular intervals for integration with third-party applications. It is provided here solely to illustrate how results appear in a real-world scenario. **Please note** that indentation has been added to this example for readability purposes.

```
{
  "CurrentScanPhase" : 6,
  "CurrentScanPhaseName" : "Filesystem Objects",
  "LastScanPhase" : 9,
  "ItemsScanned" : 100218,
  "PUCount" : 1,
  "VirusCount" : 0,
  "ScanCompletion" : 22,
  "CurrentlyScannedFile" : "C:\WINDOWS\FONTS\SCRIPT.FON",
  "CurrentVirus" : "",
  "CurrentPU" : "PUP.Optional.DotPitch",
  "ElapsedTime" : "00:00:34:840"
}
```

Event Logging to syslog

Breach Remediation integrates very easily into a corporate network, providing highly effective results in the detection and remediation of threats on endpoints. That integration has been extended further through the addition of event logging using industry-standard methods. Based on user requests, we have implemented logging using CEF (Common Event Format), and more specifically, output is tailored to the ArcSight Security Intelligence platform and others which support the CEF format.

This section of the guide is devoted to detailed descriptions of how we have implemented event logging, so that you may easily understand log results and customize reporting in your specific environment.


Construction of a Log Entry

All event logs use a standardized format, which consist of an external logger prefix, a header and an extension. They are described as follows:

- **syslog prefix:** A mandatory entry that is applied for compliance with syslog standards. It includes:
 - **Event date**, including month, day and year, in the format (e.g., **Jul 15 2021**)
 - **Event time**, using 24-hour clock, in the format **hh:mm:ss** (e.g., **12:25:40**)
 - **Hostname** which logs pertain to (e.g., **SFO-VM1234.internal.contoso.com**)
- **Header:** Mandatory fields which identify the product/client generating log entries. Vendors may use non-standard field names for these fields, but their usage must correspond to fields and their order within the log record.
 - **CEF Version**, in the format “**CEF:<version>**”. **<version>** is a single digit and is used for compliance with the CEF standard as well as to specify how remaining data should be interpreted.
 - **Device Vendor** identifies the vendor of the product/client which is generating log entries.
 - **Device Product** identifies the product/client which is generating log entries.
 - **Device Version** identifies the product/client version. Breach Remediation is identified not only by the version of the executable, but also by each major components used in conjunction with the program. All components which follow the executable program version are bounded by square brackets. Those components are:
 - Engine (MBAM Core DLL)
 - Rules Database
 - Actions Database
 - Anti-Rootkit Database (Swiss Army Knife)
 - **Signature ID** is a unique numeric identifier which is assigned to each event type. A full list of all Signature IDs can be found later in this section.
 - **Name** is a simple text description for each event that corresponds to a specific Signature ID
 - **Severity** is the relative importance of any event, with 1 being the least important and 10 representing a critical event.
- **Extension:** This is a series of fields which are not mandatory by CEF standards. These fields represent values that each vendor select for inclusion in their event logs. Because these fields are not mandatory, vendors will commonly use non-standard field names, and may also include labels for the non-standard fields. If a customer elects to use these labels, this would improve readability of log information, though the same label used by multiple vendors may also create confusion for the user.

Mapping Fields to CEF Format

As mentioned previously, log entries are comprised of three separate sections. The *syslog prefix* and *Header* are mandatory and must conform to rigid standards. The *extension* provides flexibility that vendors require to capture important details related to their products/clients, while still conforming to CEF standards.

CEF Field Usage			
sorted by CEF Standard Field Name			
CEF Standard Field Name	Malwarebytes Field Name	Type	Description -or- Explicit Value
CEF Header			
deviceEventClassId	EventId	integer	Event type
deviceProduct	ProductName	string	Product name
deviceVendor	Company	string	Product vendor's name
deviceVersion	ProductVersion	string	Product version
Name	EventName	string	Event name
Severity	Severity	integer	Severity (1=min, 10=max)
CEF Extension 			
act	Action	string	Action taken with regard to malware
cat	MalwareCategory	string	Either "pu" or "virus"
cs1	MalwareName	string	Name of detected malware
cs2	MalwareHash	string	MD5 hash of detected malware
cs3	SessionId	string	UUID for each MBBR session
cs4	MalwareClass	string	Identifies object type containing malware
cs5	CommandLine	string	Command with arguments executed by user
deviceMacAddress	MACAddress	MAC	MAC address of host where MBBR runs
dvchost	Hostname	string	Hostname where MBBR runs
end	DateTime	time	Event End Date/Time
filePath	FilePath	string	Location of detected malware
msg	*	string	Multi-purpose text string
outcome	Result	string	"succeeded", "failed" Scan commands also allow "stopped", "cancelled"
rt	DateTime	time	Event Date/Time
start	DateTime	time	Event Start Date/Time
suser	UserName	string	Name of user who runs MBBR

As mentioned previously, *msg* is a multi-purpose field. The CEF format provides six fields which may contain custom data, that which the vendor has determined to be important with relation to their product/client, but does not conform to a standard CEF field. We utilize five of these six fields, and also utilizes *msg* to provide more robust log content. Its usage in each log message will be detailed in the next section.

Log Events

Breach Remediation currently generates log entries for nine different event categories. This section of the guide describes each of those categories in detail. The following table lists fields which are common to all log entries created by *Breach Remediation*.

CEF Standard Field Name	Malwarebytes Field Name	Description -or- Explicit Value
CEF Header		
deviceVendor	Company	"Malwarebytes"
deviceProduct	ProductName	"Malwarebytes Breach Remediation"
deviceVersion	ProductVersion	Version of program, engine and databases
CEF Extension		
cs3	SessionId	UUID for each MBBR session
dvchost	Hostname	Hostname where MBBR runs
deviceMacAddress	MACAddress	MAC address of host where MBBR runs
cs5	CommandLine	Command with arguments executed by user
outcome	Result	"succeeded", "failed" Scan commands also allow "stopped", "cancelled"
suser	UserName	Name of user who runs MBBR

In addition to these common fields, the following events utilize several fields specific to the event being logged. The remainder of this section is devoted to descriptions of each of these events.

1000 – ScanStartEvent

1000	ScanStartEvent	Generated when a scan command is initiated		
	EventName	Severity	Fields	Mapping
	ScanStarted	1	ScanType (string)	msg Format: "msg=ScanType:%s" Value(s): threat hyper path full
			Time	start

1001 – DetectionEvent

1001	DetectionEvent	Generated when malware is detected during a scan		
	EventName	Severity	Fields	Mapping
	Malware Detected	9 (PUM) 10 (virus)	Action	act Value(s): none
			MalwareCategory	cat Value(s): pu (PUM) virus (virus malware)
			MalwareName	cs1
			MalwareHash	cs2
			MalwareClass	cs4 Value(s): 0x01 (File) 0x02 (Folder) 0x04 (RegKey) 0x08 (RegVal) 0x10 (Process) 0x20 (Module) 0x40 (Ads) 0x80 (Physical Sector)
			Time	real-time

1002 – RemovalEvent

1002	RemovalEvent	Generated when malware is removed during "scan -remove" or "scan -removelastscan" commands		
	EventName	Severity	Fields	Mapping
	Malware Detected	9 (PUM) 10 (virus)	Action	act Value(s): quarantined
			MalwareCategory	cat Value(s): pu (PUM) virus (virus malware)
			MalwareName	cs1
			MalwareHash	cs2
			MalwareClass	cs4 Value(s): 0x01 (File) 0x02 (Folder) 0x04 (RegKey) 0x08 (RegVal) 0x10 (Process) 0x20 (Module) 0x40 (Ads) 0x80 (Physical Sector)
			Time	real-time

1003 – ScanEndEvent

1003	ScanEndEvent	Generated when a scan command ends		
	EventName	Severity	Fields	Mapping
	ScanEnded	1	DetectionCount(int)	msg
			RemovalCount(int)	msg Format: "msg=DetectionCount:%d Removal Count:%d" Note: Fields are combined as part of the <i>msg</i> field
			Time	end

1004 – RestoreStartEvent

1004	RestoreStartEvent	Generated when a "quarantine -restoreall" command is initiated		
	EventName	Severity	Fields	Mapping
	Restore Started	1	Time	Start

1005 – RestoreEvent

1005	RestoreEvent	Generated when an item is restored during a "quarantine -restoreall" command		
	EventName	Severity	Fields	Mapping
	Item Restored	5	Action	act Value(s): restored
			FilePath	filePath Value(s): Name and complete path of a file being restored
			Time	real-time

1006 – RestoreEndEvent

1006	RestoreEndEvent	Generated when a "quarantine -restoreall" command ends		
	EventName	Severity	Fields	Mapping
	RestoreEnded	1	DetectionCount(int)	msg
			RestoreCount(int)	msg Format: "msg=RestoreCount:%d"
			Time	end

1007 – RemoveLastScanEvent

1007	RemoveLastScanEvent	Generated when a "scan -removelastscan" command is initiated		
	EventName	Severity	Fields	Mapping
	RemoveLastScan	1	RemovalCount(int)	msg Format: "msg=Removal Count:%d"
			Time	real time

1008 – DbUpdateEvent

1008	DbUpdateEvent	Generated when a "update" command is initiated		
	EventName	Severity	Fields	Mapping
	Database Rules Update	1	EngineVer (String)	msg
			RuleVer (string)	msg
			SwissArmyVer (string)	msg
			ActionVer (string)	msg
			Time	real time
				Format: "msg=EngineVer:%s RuleVer:%s SwissArmyVer:%s ActionVer:%s" Note: Fields are combined as part of the <i>msg</i> field

1009 – CustomDbUpdateEvent

1009	CustomDbUpdateEvent	Generated when the "settings" command on -customdb.<xxx> option is invoked. NOTE: <xxx> is add, load, clear or enabled:true false		
	EventName	Severity	Fields	Mapping
	Custom Db Rules Update	1	Action (str)	msg
			RulesAdded (int)	msg
			RulesIgnored (int)	msg
			Time	start
				Format: "act=%s msg=RulesAdded:%d RulesIgnored:%d" Note: Fields are combined as part of the <i>msg</i> field

Creating Custom Rules

Custom rules can be specified through a json file. While Mandiant's IOC Editor provides the capability to create a wide range of rules for identifying malware, five specific rules apply to *Breach Remediation*. For each of these rule types, an example is provided to show the construction of the rule, content examples, applicable criteria, and when the rule applies (relative to the scope of *Breach Remediation*).

Custom Hash Rule

The Custom Hash rule provides for identification of a threat using its 32-bit MD5 hash value.

Syntax:

```
{
  "condition" : "is",
  "md5" : "3BAA69B0C31CA2CCFB30D75A83B060AA",
  "name" : "TestThreat.CustomHashRule",
  "type" : "hash"
}
```

Criteria:

Condition value = "is"

When Used:

Full scan, Hyper scan, Threat scan, Path scan

Custom File Rule

The Custom File rule identifies a file (by name).

Syntax:

```
{
  "condition" : "is",
  "fileName" : "ThreatFile.exe",
  "name" : "TestThreat.CustomFileRule.Is",
  "type" : "file"
}

{
  "condition" : "contains",
  "fileName" : "ThreatFile",
  "name" : "TestThreat.CustomFileRule.Contains",
  "type" : "file"
}
```

Criteria:

Condition value = "is" or "contains". When "is" condition is used, the filename does not include the directory path. The filename is case-insensitive.

When Used:

Full scan, Hyper scan, Threat scan, Path scan

Custom Folder Rule

The Custom Folder rule identifies a folder/path.

Syntax:

```
{
  "condition" : "is",
  "folderPath" : "C:\\abc\\ThreatFolder",
  "name" : "TestThreat.CustomFolderRule.Is",
  "type" : "folder"
}

{
  "condition" : "contains",
  "folderPath" : "ThreatFolder",
  "name" : "TestThreat.CustomFolderRule.Contains",
  "type" : "folder"
}
```

Criteria:

Condition value = "is" or "contains". When "is" condition is used, the folder path is the absolute path of the folder. The folder path is case-insensitive.

When Used:

Full scan, Hyper scan, Threat scan, Path scan

Custom Registry Key Rule

The Custom Registry Key rule identifies a specific registry key. Please note that RegistryItem/Path and RegistryItem/KeyPath are treated the same by *Breach Remediation*.

Syntax:

```
{
  "condition" : "is",
  "keyPath" : "HKLM\\abc\\ThreatRegKey",
  "name" : "TestThreat.CustomRegKeyRule",
  "type" : "regkey"
}
```

Criteria:

Condition value = "is". The registry key is case-insensitive.

When Used:

Full scan, Threat scan

Custom Registry Value Rule

The Custom Registry Value rule identifies a specific registry value. It requires three (3) child indicator items to properly identify the registry value.

Syntax:

```
{
  "condition" : "is",
  "keyPath" : "HKLM\\abc\\ThreatRegKey",
  "name" : "TestThreat.CustomRegValueRule",
  "type" : "regvalue",
  "valueData" : "sampledata",
  "valueName" : "name1"
}
```

Criteria:

All indicators should use "is" condition. All indicator values are case-insensitive.

When Used:

Full scan, Threat scan

Program Status Codes

In the course of operation, *Breach Remediation* returns a status code for each command that has been executed. Environmental variable `errorlevel` is used for this purpose. Status codes are in hexadecimal notation. Status code 0 (zero) represents successful completion of the requested command, while all non-zero values represent failures. File `mbberr.h` (located in the `./Doc` subdirectory) contains a full listing of all status codes.

Further reading

We recommend that you obtain a copy of the following document from ArcSight. It is a detailed guide pertaining to the CEF logging format, as well as their recommendations targeted to users and developers.

- ***Implementing ArcSight CEF*** (Revision 20, dated 05 June 2013)
<https://protect724.hp.com/docs/DOC-1072>
- ***Mandiant ICO Editor User Guide*** (version 2.2.0.0)
<https://www.fireeye.com/content/dam/fireeye-www/services/freeware/ug-ioc-editor.pdf>
- OpenIOC web site
<http://www.openioc.org/>

Appendix A: Demo Examples

The following examples are taken from our Demo Guide and should prove useful.

Console settings

```
mbr settings -color:on  
mbr settings -color:off
```

Error Log settings

```
mbr errorout -errlog:c:\temp\Errlog_123.txt  
mbr errorout -reset  
mbr errorout -console:on  
mbr errorout -console:off
```

Proxy settings

```
mbr settings -proxy.clear  
mbr settings -proxy.port:8000  
mbr settings -proxy.server:10.151.110.149  
mbr settings -proxy.user:admin  
mbr settings -proxy.password:Test!234  
mbr settings -proxy.useauth:true  
mbr settings -proxy.enabled:true
```

Verify settings

```
mbr settings -proxy.test  
mbr settings -proxy.enabled:true  
mbr settings -proxy.clear  
mbr settings -proxy.port:8000  
mbr settings -proxy.server:10.151.110.149  
mbr settings -proxy.test
```

Single line (“mbr settings” is only needed once)

```
mbr settings -proxy.port:8000 -proxy.server:10.151.110.149 -proxy.user:admin -proxy.password:Test!234 -  
proxy.enabled:true  
mbr settings -proxy.server:xyz.com -proxy.port:80 -proxy.enabled:true  
mbr settings -proxy.port:8000 -proxy.server:10.151.110.149 -proxy.user:admin -proxy.password:Test!234 -  
proxy.enabled:true
```

Other Single line Examples

```
mbr register -key:#####-#####-#####-##### && mbr update && mbr scan -full  
mbr settings -log.server:127.0.0.1 -log.port:514 -log.events:DbUpdateEvent:off -log.enabled:false  
mbr settings -log.server:127.0.0.1 -log.port:514 -log.events:DbUpdateEvent:on -log.enabled:false
```

Syslog Settings

```
mbr settings -log.enabled:true  
mbr settings -log.server:10.151.110.149  
mbr settings -log.port:514  
mbr settings -log.test  
mbr settings -log.enabled:true -log.server:xyz.com -log.port:123  
mbr settings -log.clear
```

Reset All Settings to Defaults Example 1

- 1. Changes some settings**
mbr settings -color:off
mbr settings -useStaticUrls:true
mbr settings -log.enabled:true -log.server:xyz.com -log.port:123
- 2. Reset the settings**
mbr settings -resetAll
- 3. Verify all settings were reset to the defaults**
mbr settings
- 4. Test multiple settings**
mbr settings -resetAll -log.enabled:true -log.server:xyz.com -log.port:123

ResetAll Settings to Defaults and Syslog allevents OnOff

```
mbr settings -resetAll  
mbr settings -log.events:allevents:on|off
```

Testing

```
mbr settings color:off  
mbr settings -resetAll  
mbr settings -log.allevents:off  
mbr settings -proxy.server:xyz.com -proxy.port:80 -proxy.enabled:true  
mbr settings -useStaticURLs:true  
mbr settings -log.events:ScanEndEvent:on  
mbr settings -log.events:ScanStartEvent:on  
mbr settings -resetAll
```

Set/Clear all Syslog Examples

```
mbr settings -log.AllEvents:off  
mbr settings -log.events:filecreated:on  
mbr settings -log.AllEvents:on
```

Set Syslog protocol to TCP/UDP (TCP is the default.)

```
mbr settings -log.netprotocol:tcp  
mbr settings -log.netprotocol:udp
```

Excludelist using staticURL

```
mbr scan -excludelist:https://url.com
```

Excludelist using simple local file

```
mbr scan -threat -excludelist:xlist2.json  
mbr scan -hyper -excludelist:sampleExclusions.json
```

Exclusions

```
mbr scan -threat -exclude:"c:\temp"  
mbr scan -threat -excludelist:xlist2.json  
mbr scan -path:c:\temp2 -exclude:c:\temp2\TestFiles_32  
mbr scan -full -exclude:c:\temp33;c:\temp\
```

Syslog Test 1 (to an external syslog)

```
mbr settings -log.server:10.151.110.149
mbr settings -log.port:514
mbr settings -log.events:DbUpdateEvent:off
mbr settings -log.test
```

Syslog Test 2 (to local syslog)

```
mbr settings -log.server:127.0.0.1
mbr settings -log.port:514
mbr settings -log.events:DbUpdateEvent:off
mbr settings -log.test
mbr settings -log.enabled:true
mbr scan
```

CustomDBRules

```
mbr settings -customdb.add:<label>:<MD5HashValue>
Steps
mbr register -key:#####-#####-#####-#####
mbr settings -customdb.enabled:true
mbr settings -customdb.add:"TestThreat1:3BAA69B0C31CA2CCFB30D75A83B060AA"
mbr scan -threat
mbr update
mbr scan -threat
mbr settings -customdb.enabled:false
mbr settings -customdb.add:"TestThreat2:3BAA69B0C31CA2CCFB30D75A83B060AB"
mbr settings -customdb.add:"TestThreat3:3BAA69B0C31CA2CCFB30D75A83B060AC"
mbr settings -customdb.enabled:true
mbr update
```

ForceRun Option

Used to Detect MBAM or ARW installed and running.

***Note: Delete Logs, to make it easy to verify/compare...*

```
mbr register -key:#####-#####-#####-#####
mbr update
mbr settings -forcerun:true (Current Default, and still Default)
mbr scan -path:c:\temp\TestFiles_32
mbr settings -forcerun:false
mbr scan -path:c:\temp\TestFiles_32
**Quit MBAM or ARW.... (Should block)
mbr settings -forcerun:true (Current Default, and still Default)
mbr scan -path:c:\temp\TestFiles_32
```

Rootkit Testing

```
mbr scan -ark -path:c:\
```