URL Filtering/IP Reputation - DP Functional Specification

VeloCloud Engineering

Exported on 11/20/2024

Table of Contents

Go to start of metadata[1]

Change Log

Version	Date	Changes	Author
0.1	15 May 2023	Initial draft.	
0.2	07 Aug 2023	Updated Registration & Memory consumption	@Ramprasath G R
0.3	21 Aug 2023	Updated with security profile related changes	

Approvals

Title	Approver	Date

STATUS: APPROVED

FC Guidance

Release & Software Development Life Cycle[2]

Author(s)	DP/CP: @Bhuvaneswar Doni @Abhijeet Bhoyar @Unknown User (bsamy) @Ramprasath G R VCO:

1 https://confluence.eng.vmware.com/display/VCDP/OSPFv3+on+an+Edge#page-metadata-start
2 https://confluence.eng.vmware.com/pages/viewpage.action?pageId=621036619

Approver(s)	✓ @Adam Schultz ☐ @Gurudutt Maiya Belur ☐ @Unknown User (qxinzhou) ✓ @Preethi Nandakumar ✓ @Sathya Thammanur ☐ @Sivabalan Pandian ☐ @Kartik Kamdar ✓ @Gopal Rajagopal
Reviewer(s)	@Gopal Rajagopal @Aditya Agarwal @Ramprasath G R @Karthik Paramasivan @Unknown User (jkalaiselvan) @Qing Li @Ben Shapero @Ritesh Tiwari @gezhumalai
Status	APPROVED
Dev lead/ Architect	DP: MP: @Aditya Agarwal
DP contact	@Preethi Nandakumar
MP contact	@Unknown User (qxinzhou)
QE contact	@Preethi Nandakumar
Target Release	Yamazaki MR
PRD	Edge Firewall ATP services (IDS/IPS, URLF, Reputation)[3]
Aha Link	
Jira Epic	VLENG-119074[4] - Edge Firewall ATP services - URL Filtering & IP Reputation FIXED NOT VERIFIED
Project Page	Edge Firewall EFS services (URL Filtering, IP Reputation) Project Page - Phase 2[5]

3 https://confluence.eng.vmware.com/pages/viewpage.action?pageId=1481319163&src=contextnavpagetreemode
4 https://vmw-jira.broadcom.net/browse/VLENG-119074?src=confmacro
5 https://vmw-confluence.broadcom.net/display/VCDP/Edge+Firewall+EFS+services+%28URL+Filtering%2C+IP+Reputation%29+Project+Page+-+Phase+2

References	

1 Introduction

This section should minimally address the following

- ☐ **What's the purpose of this feature? Why do we need this?**
- ☐ **What is the problem/gap that we aim to address with this feature?**
- ☐ **How will we measure that we have achieved them?**
- ☐ **If this is an improvement of an existing feature, what are the problems with the existing feature?**
- ☐ **Highlight the primary goals and the secondary goals of this project.**
- ☐ **Are there any infrastructure goals, if this is not an infrastructure feature?**

This feature enhances velocloud edges to perform URL category and reputation based filtering and IP reputation based filtering for traffic streams that are originated from branches towards the internet. The threat intelligence related to URL/IP filtering reputation are to be sourced from VMware NSX Threat Intelligence Cloud Service aka., NTICS in the form of URL and IP Database files. These files contain information about URL category and URL/IP reputation from Webroot.

The Webroot® Threat Intelligence Platform, uses a big data architecture to provide the most comprehensive and accurate threat intelligence available today, including up-to-the-minute intelligence on IP addresses of emerging threats. This intelligence can be used to block traffic from TOR (the onion routing) nodes, proxies, botnets, and other malicious actors.

There filtering services can be configurable to monitor and/or take action.

1.1 Dependencies

This section should minimally address the following

- ☐ **Who are the various stakeholders for reviewing this document? QA is an obvious stakeholder. Who are the architectural/OPS/PM reviewers?**
- ☐ **Does this feature have Data Plane dependencies?**
- ☐ **Does this feature impact the way Ops (TechOps or CloudOps or DevOps) do their work?**
- ☐ **Any dependencies on MP/UI/Ux/Platform.**
- ☐ **Any Third party SDK/tool dependencies (DPDK/FRR/Qosmos etc..)**

This feature has UI as well NTICS depedencies.

NTICS:

NSX team provides REST API endpoints to download URL category lists and reputation databases. Unavailability of these APIs will affect ability of VCE to provide URL Filtering and IP Reputation. The webroot SDK integrated with edged process shall invoke the APIs to pull URL/IP DBs, real-time updates and perform asynchronous network query for URL categorisation.

MP:

The management plane should support the configuration of security profiles with configuration for each EFS engine and associating it with firewall rules. Management plane also should be responsible to provide the license key whenever EFS feature is enabled for the enterprise.

1.2 Risks

This section should minimally address the following

- ☐ **Does this require Program Management or Product Management to communicate to Customers beforehand?**
- ☐ **Are there significant risks/unknowns that Leads/Engineering Managers need to be aware of, like EoL components etc.?**

#	Title	Probability	Impact	Mitigation plan

1. NTICS capability of handling simultaneous requests from scaled number of edges
2. Full URL filtering is only possible with non-SSL traffic. As penetration of TLS 1.3 increases across the Internet over the next 12 months, certain elements of SNI used for domain-name filtering may no longer be usable. This limitation shall be appropriately conveyed to the customer.

1.3 Open Issues

This section should minimally address the following

- ☐ **List requirements that are ambiguous or need further clarity**

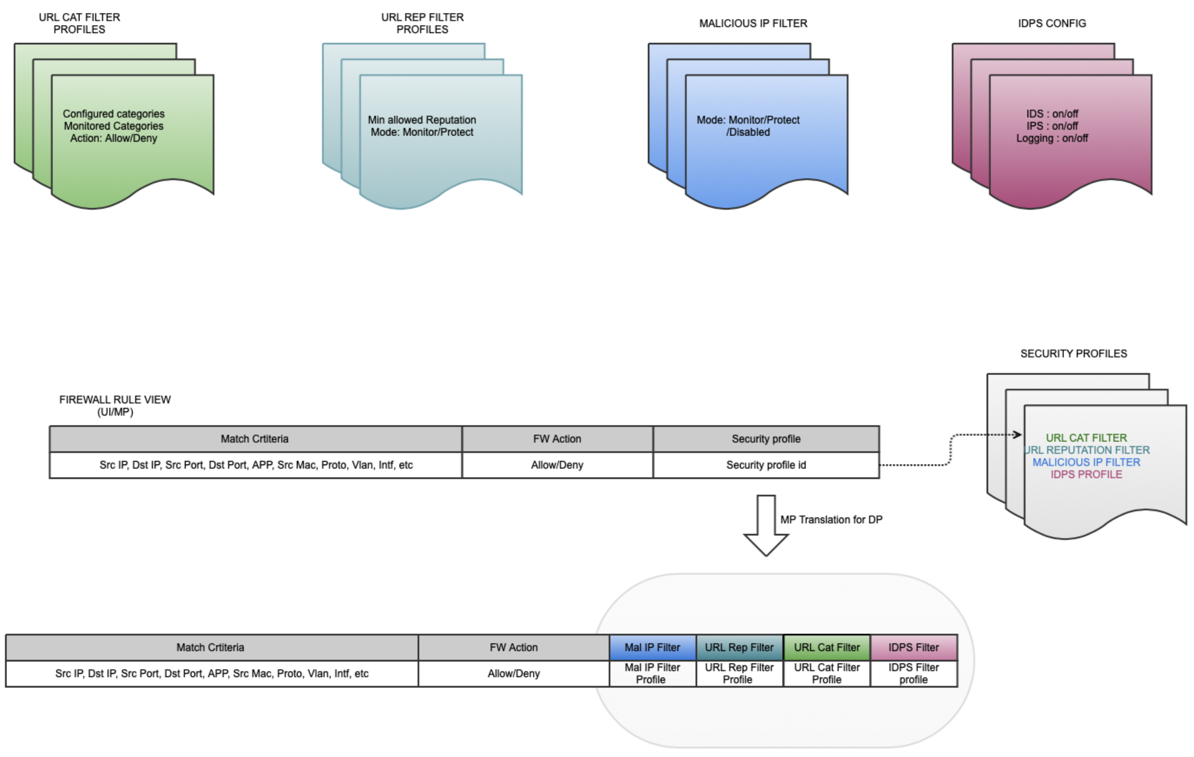
#	Title	Detail	AI Owner	Due Date

2 Functional Overview

This section should minimally address the following

- ☐ **Configurations workflow related to this feature.**
- ☐ **Have separate sections for VCO, Platform, Control Plane and Data Plane changes.**
- ☐ **Describe the high-level functional behavior of this feature.**
- ☐ **What does QA/SE/customer/partner need to know to start using this feature?**
- ☐ **What are the assumptions?**
- ☐ **What can this feature *not* do?**
- ☐ **Does this feature improve upon an existing behavior? If so, explain?**
- ☐ **Can this feature be turned off using a feature flag? If not, explain why?**
- ☐ **Is the feature consumable? If exposed to the end customer, is it easy to use? Can the user start using it quickly and/or write some automation against it easily?**

Functionally EFS is classified into 4 filtering engines and each of the engines can use preconfigured reusable profile objects to take actions based on a security profile context that a rule is associated.



Rule: Any rule will can be mapped to mapped a security profile when allow action is set.

Security profile: Security profile is a reusable object consisting of configuration mapping for each of the EFS engine. These configurations shall be made available in the form of reusable config profile objects.

Config profile objects: Every engine in EFS shall have a set of preconfigured profiles. These profiles are configured with the config parameters as seen in the figure.

2.1 Translation fo DP

Each rule will have embedded set of actions packed inside the rule blob. When EFS is enabled and a EFS security profile is associated with a rule, Management plane internally translates

3 Requirements

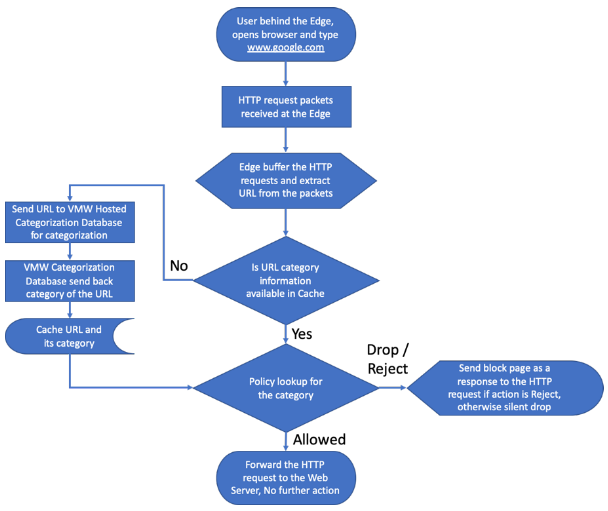
This section should cover the engineering requirements mapped to the PRD requirements.

- [ ] **List out VCO, Platform, Control Plane and Data Plane requirements mapping to PRD requirements in the form of Requirement traceability matrix.**
- [ ] **Cover the scale, performance, system and automation requirements.**
- [ ] **Have a cross reference link to PRD and FS.**

3.1 Management Plane

S.No	PRD Requirement	PRD section ID	FS section ID	Release Name

3.2 Data Plane

S.No	PRD Requirement	PRD section ID	FS section ID	Release Name
1.	**URL Filtering** For every incoming HTTP REQUEST packets, the Edge platform must extract the URL from the HTTP packets and categorize the URL (e.g., www.cnn.com[6] is a News category). Then the “allow,” "drop" and “reject” decision is made based on the URL category. A sample sequence of activities at the Edge is defined in the below figure  User behind the Edge, opens browser and type www.google.com HTTP request packets received at the Edge Edge buffer the HTTP requests and extract URL from the packets Is URL category information available in Cache No Send URL to VMW Hosted Categorization Database for categorization VMW Categorization Database send back category of the URL Cache URL and its category Yes Policy lookup for the category Drop / Reject Send block page as a response to the HTTP request if action is Reject, otherwise silent drop Allowed Forward the HTTP request to the Web Server, No further action For cache miss on local database lookup, the url will be classified as unknown category and action will be taken based on the definition for unknown categories. Query for URL category will happen asynchronously to not impact performance.	URL 1.1	4.1.3, 4.1.7, 5.1.7, 5.1.9, 5.1.10	
2.	**HTTP Protocols Support** The solution must intercept and extract URL information from the HTTP methods – GET, POST, PUT, HEAD, OPTIONS, TRACE, DELETE, CONNECT	URL 1.2	4.1.3, 5.1.9	

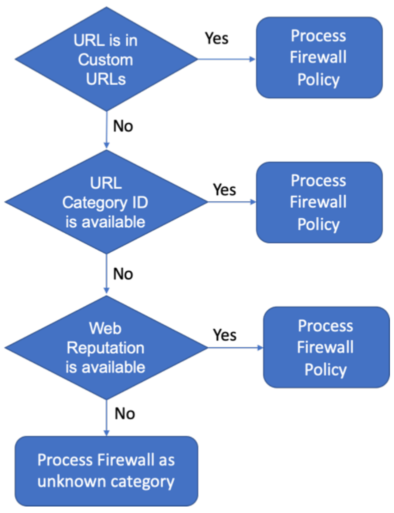
6 http://www.cnn.com/

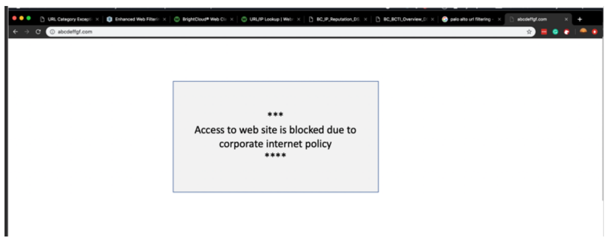
S.No	PRD Requirement	PRD section ID	FS section ID	Release Name
3.	**HTTPS Traffic Support using SNI** In the HTTPS packets, the URL information is encrypted, and it is challenging to extract the URL information from the HTTP payload without Edge acting as SSL Forward Proxy (SSL man-in-middle). Implementing and configuring the SSL proxy is not trivial. Instead, Edge platforms must snoop the SSL transaction between client and server, the SNI field in the SSL transaction provides the URL information. So, by extracting the URL information from the SNI field, the Edge platform will be able to identify the URL category	URL 1.3	4.1.3	
4.	**URL Filtering for Underlay and Overlay Traffic** URL filtering must support on the traffic going over • DMPO • non DMPO tunnels • Direct internet access WAN links (underlay)	URL 1.4	Implicit. Not to be tracked in RTM. There is no special effort.	
5.	**URL Group** URL Group is an object with a set of URL categories. A URL Group can contain the pre-defined URL categories and custom URL List. Admins can create multiple URL Groups, but only one URL Group can be referenced in one single firewall policy. A new tab must be added to the Object Groups page for the URL Groups under Security section. Refer to slides[7] for details. A new "URL Group" must-have the following sections - Reputation List, Custom URL List and Pre-defined URL Categories (arranged in alphabet order). Admin can add URLs to the Custom URL List OR select one or more pre defined URL categories into the URL Groups.	URL 1.5	NA	

7 https://onevmw.sharepoint.com/:p:/r/teams/velo-products/Shared%20Documents/SASE%20Security/edge-firewall/FW%20Security%20Groups.pptx?d=w27afba9729514c319f83f3c1a1e73d2a&csf=1&web=1&e=TYkBvA

S.No	PRD Requirement	PRD section ID	FS section ID	Release Name
6.	**Pre-defined URL Categories** There are hundreds of millions of websites and URLs. It is very tedious to configure the policy for individual URLs, so these URLs are mapped to a specific category, and then filtering policy is applied over the categories. Every URL is mapped to the specific URL category (e.g. www.cnn.com[8] is part of the news category). Except for the custom URL list, all of the URL Filtering policy decision is based on the URL category The number of URL Categories on the Edge platforms must be the same as in the Cloud-hosted Categorization Database, which in turn should match the Webroot database. As per the Webroot datasheet, Webroot has 10 URL Category Groups and 80 URL Categories.	URL 1.6	4.1.9	
7.	**Web / IP Reputation** Web / URL reputation provides the trustworthiness of the Web site. Most popular URL Filtering solution (like Webroot) track the websites overtime for the malicious and inappropriate content and assign a score between 0 to 100. • 81-100: Trustworthy • 61-80: Low risk • 41-60: Moderate risk • 21-40: Suspicious • 01-20: High risk VMware solution must include this reputation score in the policy decision. Please refer REQ#URL 1.9 for details on how Web / IP Reputation is included in the policy decision	URL 1.7	2, 4.1.2	

8 http://www.cnn.com/

S.No	PRD Requirement	PRD section ID	FS section ID	Release Name
8.	**URL Filtering Policy Evaluation Criteria** The solution must use the Webroot database only for the categorization and web reputation, all other policy configuration and decisions reside on the VCO / Edge platforms. The “allow”, “drop” or “reject” decision is made based on the following criteria  URL is in Custom URLs Yes Process Firewall Policy No URL Category ID is available Yes Process Firewall Policy No Web Reputation is available Yes Process Firewall Policy No Process Firewall as unknown category	URL 1.9	4.1.2, 4.1.4, 4.1.15, 5.1.7	

S.No	PRD Requirement	PRD section ID	FS section ID	Release Name
9.	**URL Filtering Policy Actions** Based on the criteria listed in the URL 1.7 the Edge platforms takes one of the below actions • **ALLOW**: The HTTP REQUEST forwarded to the Website; no further action taken • **DROP**: The HTTP REQUEST is silently dropped no further action taken • **REJECT**: The HTTP REQUEST dropped, and a REJECT message page is sent as a response to the user.  *** Access to web site is blocked due to corporate internet policy **** LOG action can be enabled for each of the above options. If configured, the URL Filtering log message must be generated (please see LR 2.7 for more details)	URL 1.10	4.1.13, 5.1.7 (Reject is not supported)	
10.	**Unknown Category** Apart from predefined categories (1:1 mapping with Webroot Categories), there should be an "unknown" category to catch all URLs where Webroot categorization database does not categorize URLs and no web reputation found for it. An URL is classified into an **unknown** category when it is not categorized and also not found in the "Custom URL List".	URL 1.15	5.1.7, 5.1.10	
11.	Provide users ability to configure URL Filtering by Category and by Reputation as object groups (as highlighted in URL 1.5). Refer to slides[9] for details.	URL 1.17	2	

9 https://onevmw.sharepoint.com/:p:/r/teams/velo-products/Shared%20Documents/SASE%20Security/edge-firewall/FW%20Security%20Groups.pptx?d=w27afba9729514c319f83f3c1a1e73d2a&csf=1&web=1&e=TYkBvA

S.No	PRD Requirement	PRD section ID	FS section ID	Release Name
12.	Allow users to choose any one of the of user defined URL Category and/or URL Reputation object group(s) as part of Firewall rule creation. Refer to slides[10] for details.	URL 1.18	NA (As per design, MP will flatten the objects into set of actions per rule)	
13	**Logging for Drop / Reject URLs** The Edge must generate a firewall log (Syslog format) message when a URL is Dropped / Rejected and logging is enabled. It is recommended to send these log messages to the configured external Syslog server / SIEM.	LR 2.5	4.1.13	
14.	**Logging for Allow URLs** The Edge must generate a firewall log (Syslog format) message when a URL is **allowed** and logging is enabled. It is recommended to send these log messages to the configured external Syslog server / SIEM. Collection and aggregation of URL allow messages in VCO is only available for the demo environment.	LR 2.6	4.1.13	

10 https://onevmw.sharepoint.com/:p:/r/teams/velo-products/Shared%20Documents/SASE%20Security/edge-firewall/FW%20Security%20Groups.pptx?d=w27afba9729514c319f83f3c1a1e73d2a&csf=1&web=1&e=TYkBvA

S.No	PRD Requirement	PRD section ID	FS section ID	Release Name
15.	**Format of URL Filtering Log Message** The format of the URL Filtering log message must comply with the existing firewall log message format. And it must have below additional fields in the log message (addition to the Session-Open / Session-Deny) • Log Source = "URL Filtering" • Requested URL • URL Category Example Web Filter log Fortinet (competitor) - https://docs.fortinet.com/document/fortigate/7.2.0/fortios-log-message-reference/400992/webfilter-log-support-for-cef	LR 2.7	4.1.13	

3.3 Platform

S.No	PRD Requirement	PRD section ID	FS section ID	Release Name

3.4 Scale & Performance

S.No	PRD Requirement	PRD section ID	FS section ID	Release Name

3.5 Supportability

S.No	PRD Requirement	PRD section ID	FS section ID	Release Name

3.6 Techops

S.No	PRD Requirement	PRD section ID	FS section ID	Release Name

3.7 Engineering Requirements

3.7.1 DP Requirements

1. DP shall receive, parse and process URL Filtering config per segment
2. DP shall receive, parse and process IP filtering config per segment
3. DP shall enhance firewall table to store URLF and IP reputation config per rule
4. DP shall use the WAN side Ip for all classification purposes.
5. DP shall forward/deny traffic of particular URL classification based on configuration
6. DP shall forward/deny traffic of particular URL Reputation based on configuration
7. DP shall forward/deny traffic to particular Malicious IP based on configuration
8. DP shall determine URL up-till only domain name for https which is using TLS version < 1.3
9. DP shall determine full URL in case of plain-text http.
10. DP shall use **QOSMOS DPI** engine and achieve URL name extraction
11. DP shall download URL and IP DB from **NTICS** server
12. DP shall poll for DB update and download and it to update local DB
13. DP shall obtain license key and register with NTICS server to get client id and client secret

14. DP shall encrypt license key while storing in the local config file and decrypt during retrieval.
15. DP shall authenticate periodically with NTICS server and update valid token
16. DP shall receive licence key from VCO and register with NTICS from mgd to fetch client id and client secret.
17. DP shall download the master database file or smaller one based on edge model
18. DP shall make network query to NTICS server if URL cat is not available in local DB
19. DP shall use auth token in each query made to NTICS server
20. DP shall continue to forward uncategorized URL traffic based on explicit config.
21. DP shall log the event whenever database is updated or downloaded.
22. DP shall use webroot sdk to read/update webroot data, build database and make lookups
23. DP shall make URLF & IP reputation lookup and use it's result if URL filtering & IP reputation is configured
24. DP shall sync the DB data from active to standby.
25. DP shall sync the token to standby after periodic authentication with NTICS server
26. DP shall not communicate with NTICS while in **HA standby** mode but consume update from Active
27. DP shall sync the flow along with EFS data to standby when ever there is an update to the flow.
28. DP shall make a dummy lookup for URL in DNS query in order to launch network query before the actual traffic.
29. DP shall use default namespace/special namespace in kernel for TCP communication
30. DP shall send/receive NTICS communication packets via tunnel to namespace.
31. DP shall store the NTICS IP from initial DNS request and identify the NTICS packets for forwarding into netns.(Segment NAT entry?)
32. DP shall configure to spawn a background thread inside wrsdk for downloads and updates.
33. DP shall use asynchronous URL or IP lookup query to get reputation and category data if local lookup fails.
34. DP shall use only the exception path for URL Filtering and Malicious Ip filtering
35. DP shall upon switch over, reset all unknown category flow.
36. DP shall maintain a mapping of Domain name to flow until it is categorised.
37. DP shall age out Domain to Fc mapping entry based on thresh hold timer and mark those Fc as uncategorized.
38. DP shall have provision to operate in monitor mode without taking any action for URL Cat filtering, URL Rep Filtering and Malicious IP filtering when configured
39. DP shall combine logs from different EFS engine when more than one engine produces log for same flow.

3.7.2 MP requirements

1. VCO shall provide NTICS endpoint & license key in order to enable registration with NTICS.
2. VCO shall send config for URLF and IP reputation based filtering
3. VCO shall receive and display log event stats

<To be filled by MP>

3.7.3 Supportability Requirements

3.7.3.1 DP supportability

1. DP shall provide a way to display the configuration via debug.py command.
2. DP shall provide output of URLF/IP reputation firewall wall rule hit statistics.
3. DP shall provide command to display statistics of queries made to webroot sdk
4. DP shall provide a debug command to make a URL lookup inside edged/webroot database.
5. DP shall provide a remote-diagnostics command to make a URL lookup.

4 High level design

This section should address the following

- [ ] **Pictorial representation of all components & modules that will be involved in this feature.**
    - [ ] **Include diagrams which help detail the inter component interactions.**
    - [ ] **Details on new threads created, impact of it and component owning it with the criteria for selecting the thread priority.**
- [ ] **List out details on configuration interfaces.**
    - [ ] **VCO UI mockups / Link to Figma / Deprecation of options or screens.**
    - [ ] **Details on New API / Change in old API / Deprecation of existing API.**
- [ ] **Is there a change in the JSON blob sent from VCO to DP? If so, details on the changes.**
- [ ] **Details on the control flow path, end to end flow details on configuration change.**
- [ ] **Details on Data/Operation flows, significant events triggering this change. Comparison between old behaviour and new behaviour.**
    - [ ] **Is there a change in Packet flow introduced by this feature? If so, details on it.**
- [ ] **Details on additional memory used per component.**
- [ ] **Interactions with any existing features that are to be validated**

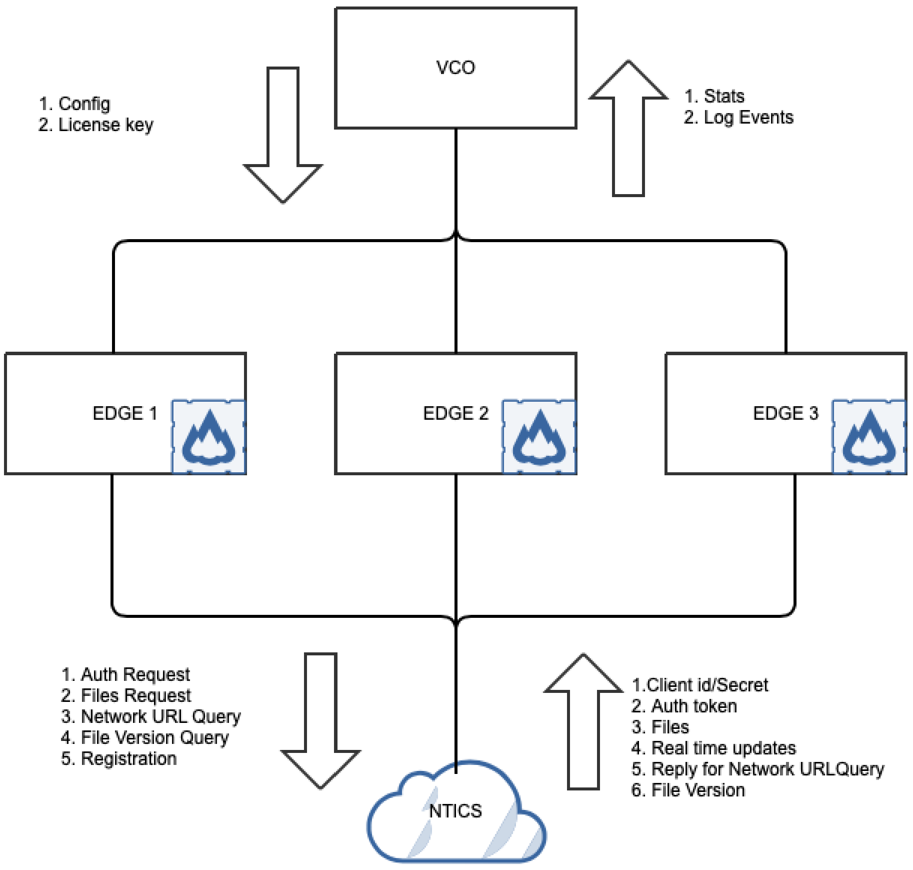
On a high level, We shall be using the databases and their updates from Webroot to determine the category of URL and reputation of URL/IP.

Accessing these databases and installing updates requires sdk from webroot. This is available as a library with APIs exposed.

When IP/URL query fails in local database, webroot sdk inside edged shall perform network query to remote NTICS server.

This communication requires a authentication which shall be explained in later sections.

DP Interaction with VCO and NTICS



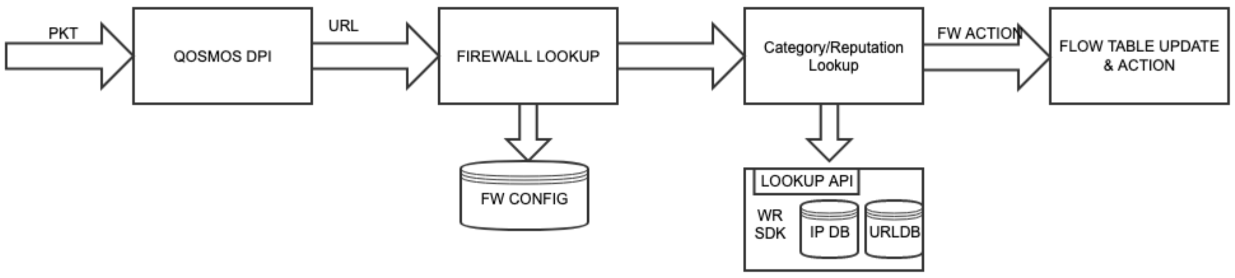
4.1 MP Design

Refer MP UI Mockups - https://www.figma.com/file/udG3bfEiaJYuUD5z0hZAVf/Edge-Firewall-ATP?type=design&node-id=2692%3A24505&mode=design&t=OqKnM1u8mA3DfcpY-1

Refer MP FS - Edge Firewall ATP - URL Filtering/ IP Filtering (Management Plane) Functional Spec[11]

4.2 Packet Exception path

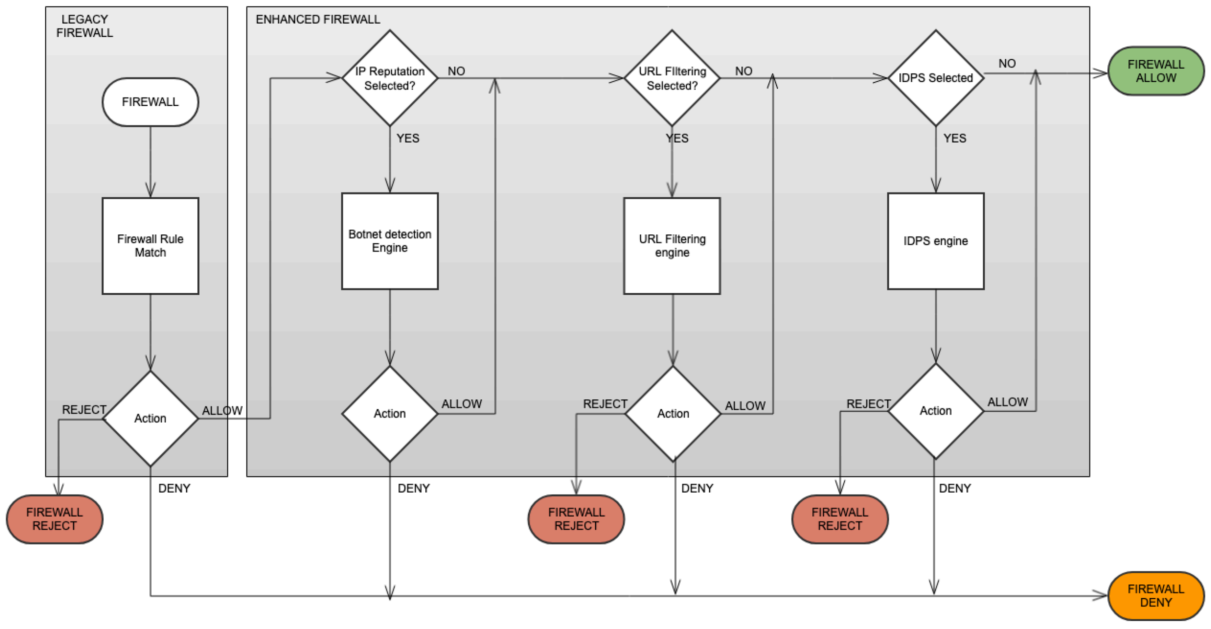
11 https://confluence.eng.vmware.com/pages/viewpage.action?pageId=1644961535



Packet that are going into DPI engine (QOSMOS) engine shall extract URL which is the key for Webroot sdk to lookup in its reputation DB.

Once found a firewall match for this key, configured action shall be honoured by subjecting to the enhanced firewall engines as per the configured security profile.

The following diagram represents the logical packet forwarding based on enhanced firewall engine decisions. Reject action is provisionally added to the spec but actual implementation shall not initially contain this option.



4.3 URL Extraction

URL extraction is done using QOSMOS DPI. Appropriate parameters in API calls are used to extract SNI incase of HTTPS (< TLS 1.3) and URL in case of HTTP plain-text traffic.

This URL is subjected in to webroot URL lookup API for obtaining category and reputation.

4.4 URL Categorization & IP reputation Database

These are separate files received from NTICS server to start with. There are two types of database files for URL categorization.

- Large database - 20M entries (400 MB)
- Small database with frequently hit URLs - 1M entries (20 MB)

IP reputation has 20 MB file.

Apart from the base files, we have option of retrieving real time updates every 5 mins.

To enhance performance and to throttle NTICS request load, we shall **consider reducing the frequency of updates.**

Usually DB files are automatically downloaded by SDK itself to the database location configured.

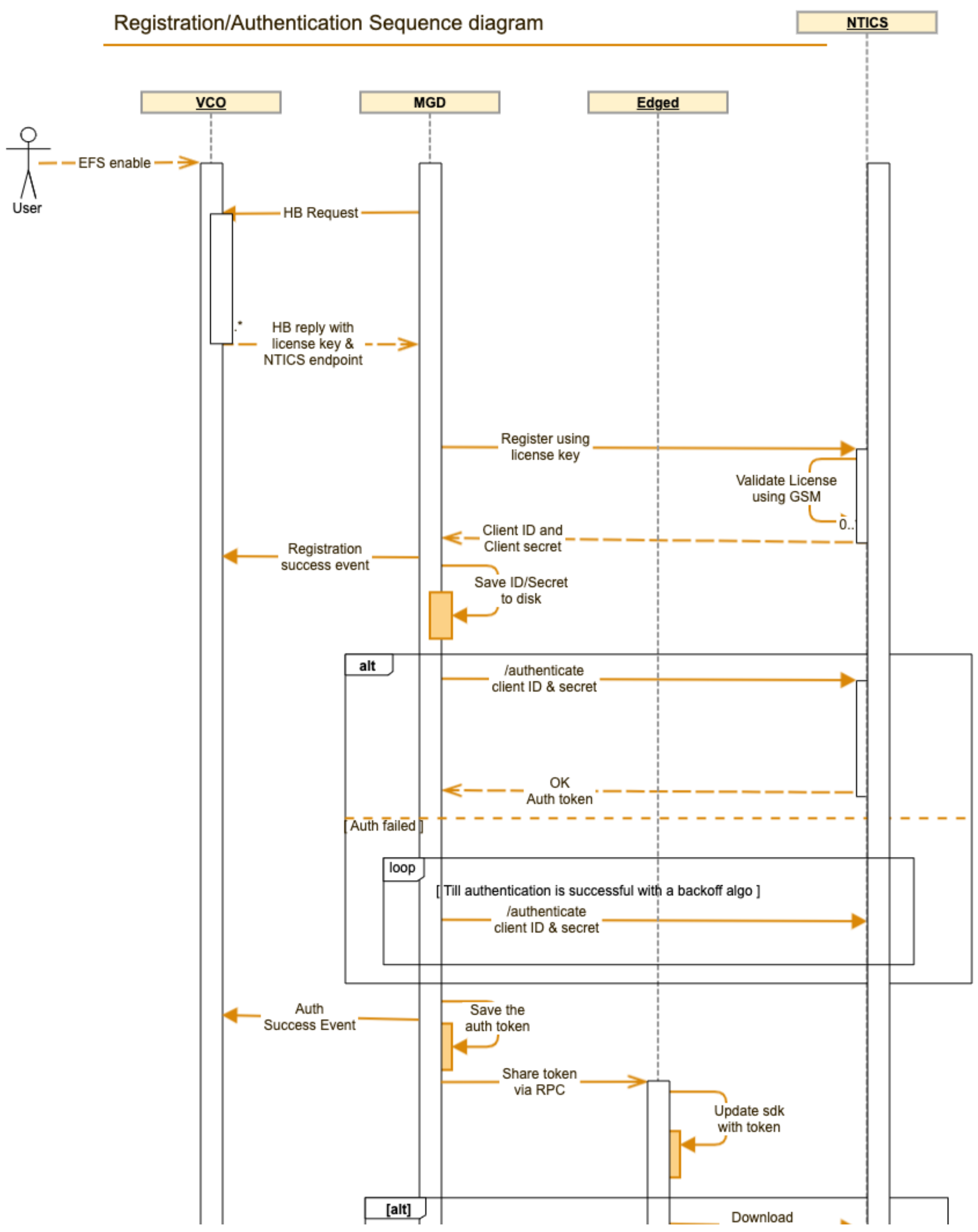
4.5 Registration with NTICS

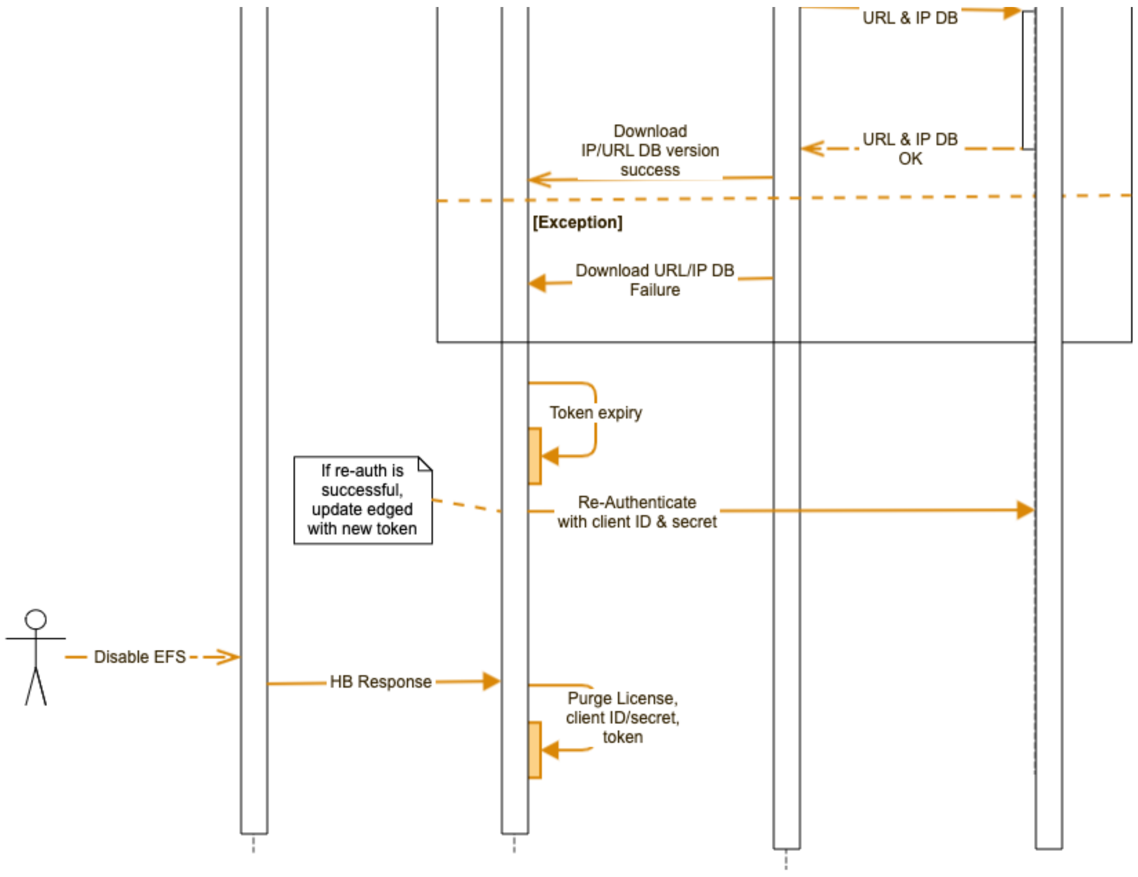
HB Response from VCO

HB Response

```
{
    "actions": [{
        "action": "configurationUpdate",
        "data": {
            "module": "atpMetadata",
            "version": "1614693596464",
            "schemaVersion": "3.0.0",
            "use": {
                "atpUpdateEnabled" : true,
                "ntics": {=====================================================> New
key added to send license information to edge
                    "licenseLogicalId": "33f8a91b-324a-11ee-9dca-0e813ba16025",
                    "licenseKey": "B7J2G-QGTF7-EUFGZ-E2DRD-UD1GM",
                    "endpoint" : "https://test-ntics.com",
                    "deviceType": "SDWAN-Edge",
                    "registerAPI": "/1.0/auth/register",
                    "authenticateAPI": "/1.0/auth/authenticate",
                    "version": "1614693596464"
                }
        }
    }]
}
```

- Edges that receive this configuration Update action make a call to ntics Endpoint and provide the license key in its request.
- NTICS validates license key and responds with client id and secret. Edges request authentication token using client id and secret to NTICS and NITCS provides auth token which will be used by edges for subsequent requests until token expires. Reauthentication shall be done before the current token expires(Proposed 30 mins reauth while the token life is 1 hour)


Registration/Authentication Sequence diagram
NTICS
VCO
MGD
Edged
User
EFS enable
HB Request
HB reply with license key & NTICS endpoint
Register using license key
Validate License using GSM
Client ID and Client secret
Registration success event
Save ID/Secret to disk
alt
/authenticate client ID & secret
OK Auth token
[Auth failed]
loop
[Till authentication is successful with a backoff algo]
/authenticate client ID & secret
Auth Success Event
Save the auth token
Share token via RPC
Update sdk with token
[alt]
Download


URL & IP DB
Download IP/URL DB version success
URL & IP DB OK
[Exception]
Download URL/IP DB Failure
Token expiry
If re-auth is successful, update edged with new token
Re-Authenticate with client ID & secret
Disable EFS
HB Response
Purge License, client ID/secret, token

Registration and Authentication of Velo edges : Before the clients can access any of the APIs provided by the cloud service, they need to register in order to get the necessary credentials.

```
Register a client with cloud cache
POST https://<cloud-svc-domain>/1.0/auth/register
{
    "license_keys": [XXXXX-XXXXX-XXXXX-XXXXX-XXXXX"],
    "device_type":"Velo-Edge",
    "client_id": "test-123"
}

Authenticate client with using secret key and client id :
POST https://<cloud-svc-domain>/1.0/auth/authenticate
{
    "client_id": "3decd544-d76c-4e42-b0f5-bdb0c16d58c3",
    "client_secret": "BdwiE6iQT8sQ/
hiACzJGZ+55zXvC32dhRapVQrrXaTMjQYKdG5UN0KxlP3nJJ0BADaWBmkZUc2iYv9FXpDR2/
JcoeWrnec8CLUJuEh35vDi45Y522B/
lpiuEaUVUnEuwxcu8lH2D7cOAb0KZa3bghUgnEsLiJN2KJVcyUw4DZWU="
}
```

On a HA system, only active will register to the NTICS. Post authentication, client-id, client-secret and authentication token shall be synced to STANDBY.

Handling Unsuccessful registration

Every unsuccessful attempts of registration will result in event generation to VCO and retry using a exponential backoff until it becomes 30 mins.

```
delay = min(base_delay * (2 ** retry_attempt), 1800)
```

4.6 Authentication

Authentication is done using http POST with client id and secret

Request:

`POST https://<cloud-svc-domain>/1.0/auth/authenticate`

Authenticate request

```
{
    "client_id": "3decd544-d76c-4e42-b0f5-bdb0c16d58c3",
    "client_secret": "BdwiE6iQT8sQ/
hiACzJGZ+55zXvC32dhRapVQrrXaTMjQYKdG5UN0KxlP3nJJ0BADaWBmkZUc2iYv9FXpDR2/
JcoeWrnec8CLUJuEh35vDi45Y522B/
lpiuEaUVUnEuwxcu8lH2D7cOAb0KZa3bghUgnEsLiJN2KJVcyUw4DZWU="
}
```

Response:
Authenticate Response

```
{
    "access_token":
"eyJhbGciOiJIUzUxMiJ9.eyJqdGkiOiIzZGVjZDU0NC1kNzZjLTRlNDItYjBmNS1ijk1MzcxNzksImlhdCI6M
TU2OTUzMzU3OX0.Pc9nvfryeZlP9CzCZLgYx3QWUaHvhjY5IKkiJLs4vLeHCfUNN4xmUzk5sbuCtmIUSJ7zYOV
AGTPJnuVXk7a-9Q",
    "token_type": "bearer",
    "expires_in": 3600,
    "scope": "[enterprise_features, distributed_threat_features]"
}
```

Edges(Mgd) shall periodically re-authenticate before expiry of token using the same POST format. Expiry time is present in response.

Every request to NTICS has to contain NCS Auth header field(`X-NCS-Authorization`) with this token.

Webroot SDK will be modified to include the header in each of those request.

4.7 Look up order & Mode

URL lookup shall be made as a synchronous call in the local database.

If not found, we shall make using asynchronous lookup API with a call back which would generate a log.

The call back shall lookup on FW config DB and appropriately update the flow.

Query order is

- RTUs
- local database file
- in-memory cache (This usually gets updated from network queries)

Cloud lookups are enabled by default. So a network query is performed if no result is found locally. Whenever a network query is replied, the result is stored in the cache. Further lookup for the same URL shall be locally queried from the cache.

4.8 Database files

URL Database and IP databases are received as files from NTICS. Webroot SDK periodically queries and requests if update version available.

These files are stored at a preconfigured location.

4.9 URL Categories

Supported categories are defined in database file. Uncategorized shall be referred with an unused Cat id . Currently supported categories are

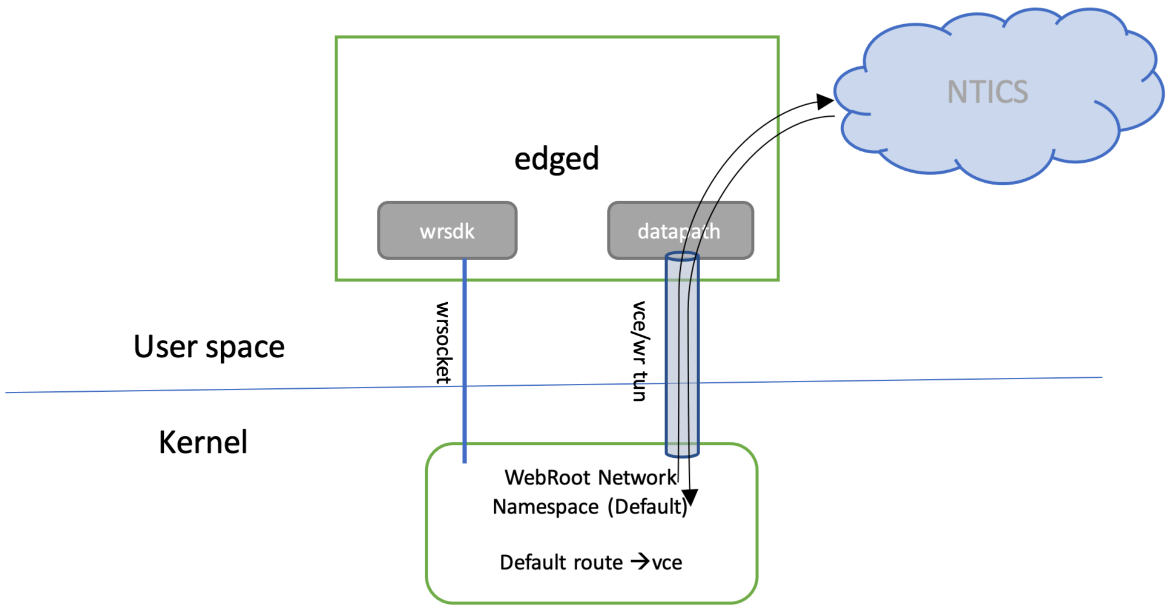
```
1       Real Estate
2       Computer and Internet Security
3       Financial Services
4       Business and Economy
5       Computer and Internet Info
6       Auctions
7       Shopping
8       Cult and Occult
9       Travel
10      Abused Drugs
11      Adult and Pornography
12      Home and Garden
13      Military
14      Social Networking
15      Dead Sites
16      Individual Stock Advice and Tools
```

```
17      Training and Tools
18      Dating
19      Sex Education
20      Religion
21      Entertainment and Arts
22      Personal sites and Blogs
23      Legal
24      Local Information
25      Streaming Media
26      Job Search
27      Gambling
28      Translation
29      Reference and Research
30      Shareware and Freeware
31      Peer to Peer
32      Marijuana
33      Hacking
34      Games
35      Philosophy and Political Advocacy
36      Weapons
37      Pay to Surf
38      Hunting and Fishing
39      Society
40      Educational Institutions
41      Online Greeting Cards
42      Sports
43      Swimsuits and Intimate Apparel
44      Questionable
45      Kids
46      Hate and Racism
47      Personal Storage
48      Violence
49      Keyloggers and Monitoring
50      Search Engines
51      Internet Portals
52      Web Advertisements
53      Cheating
54      Gross
55      Web-based Email
56      Malware Sites
57      Phishing and Other Frauds
58      Proxy Avoidance and Anonymizers
59      Spyware and Adware
60      Music
61      Government
62      Nudity
63      News and Media
64      Illegal
65      Content Delivery Networks
66      Internet Communications
67      Bot Nets
68      Abortion
69      Health and Medicine
```

```
70        Confirmed SPAM Sources
71        SPAM URLs
72        Unconfirmed SPAM Sources
73        Open HTTP Proxies
74        Dynamically Generated Content
75        Parked Domains
76        Alcohol and Tobacco
77        Private IP Addresses
78        Image and Video Search
79        Fashion and Beauty
80        Recreation and Hobbies
81        Motor Vehicles
82        Web Hosting
83        Food and Dining
85        Self Harm
86        DNS Over HTTPS
87        Low-THC Cannabis Products
88        Generative AI
999       the end
```

4.10 Webroot SDK - NTICS communication

Proposed network path for Webroot SDK ←→NTICS



With current data path setting and ip table config, http packets are forwarded to/fro Kernel default namespace where webroot http sockets are opened

Provisional design is to operate webroot traffic in separate namespace. (future consideration)

NTICS server ip is stored in a global data structure after DNS resolution. This IP is compared against incoming/outgoing packet to classify webroot control packets.

In the ingress path src ip is matched against NTICS IP and dst ip against system self ip and forwarded to Kernel using the edged mgmt tun.

In the egress path, edged receives the packet on the tunnel interface and this is queued to fast path packet processing.

4.11 DNS for traffic anticipation

Whenever edge see a DNS request from client, a dummy lookup for the URL shall be launched for obtaining URL Reputation/categorization.

This shall help in minimizing the time elapsed for network query when actual data traffic arrives, if not found in local cache.

4.12 HA Overview

Active shall sync the following with regard to URLF/IP Reputation

Mgd Sync:

- Client id and Client secret on successful registration
- Auth token shall be sync ed to standby mgd from active mgd periodically whenever a new auth token is received and that shall be updated in to standby edged from standby mgd.
- Auth Token validity info such as received timestamp and validity period is also synchronized to standby
- URLF/IP reputation config

Edge sync:

- Whenever a flow's URL/IP categorized and firewall rule is applied over it, the same state shall be synchronized to stand by along with the flow.
- Network query is not synchronized. After switchover, new active would have to start querying network again even if it was already queried by old active.

Standby on becoming Active shall check if client id and secret are available. If not, New active will re-register to get the new client id and client secret. Otherwise will just re authenticate with the client id and secret available.

Till reauth is completed, New Active shall continue to use the token synchronized from old active.

Switchover behaviour

The new active will continue to honour the firewall result synchronised from previous active.

In case of URL Cat/Reputation based filter, for the flows which has not got categorised/category synchronised at the time of switchover, any subsequent packets to these flows will cause to reset these flows.

4.13 Logging

Whenever edge encounters a flow match to a firewall policy based on IP Reputation/URL Rep/Cat filtering, we shall generate a firewall log with the action

The log shall be formatted inline to current firewall log with additional fields along the action taken.

- Log Source = "URL Filtering"/"IP Reputation"
- Requested URL
- URL Category
- URL Reputation
- IP threat

Webroot logs are stored in /velocloud/bcti.log.

Webroot logging is is not enabled by default and customer will not have option to enable this. Only support and dev can enable this. Support should enable this only if needed and only under direction from dev because this can affect performance of edge and the file can get bloated fast causing memory issues.

The reputation score classification for URL(s) and IP addresses is as given below:

Score Range	**Description**
1 - 20	High Risk
21 – 40	Suspicious
41 – 60	Moderate Risk
61 – 80	Low Risk
81 – 100	Trustworthy

4.13.1 Logging based on Engine:

IDPS, Malicious IP engine and URL category engine will be generating individual logs.

This logs will not be concatenated.

Though URL category and URL reputation URL Category and URL reputation logs will be added in single log.

If based on configuration of service group and engines enabled.

Examples : TODO

Important Notes:

1. There is no change in Legacy firewall logging.
2. There is no change in IDPS logging.
3. If an engine is not configured the respective object will not be packed.
4. Allow, Monitor and Block are the engine specific actions possible for URL Category, URL Reputation and Malicious IP Engine.
5. Monitor = Allow and Log.
6. debug.py --user_firewall_dump, edged_firewall.log (text based logging), VCO Firewall logging and VCO Monitoring logging will show these values.
7. In debug.py --user_firewall_dump for blocked traffic we will show Block. For all other logging we will use Deny.

4.13.2 Logging limitations:

1. URL == NULL and Domain == NULL in log:
   a. When there are existing flows for which there is already URL extracted and efs action is taken, if user changes URL category or URL reputation then new log if generated will have URL=NULL and Domain=NULL.
      i. There is no functionality impact here.
      ii. Only logging will have url and domain as NULL. But user can use the flow ID to look into previous logs to know the URL and Domain.
      iii. The reason for this is we don't cache DPI extracted URL and Domain information. The cost of caching this information for the sake of logging is costly
   b. Flows will generate logs in active when url and domain is present. If these flows switchover/ failover to standby (new active) only buffered webroot url and malicious ip lookup information will be buffered in new active. So we will generate logs with url=NULL and domain=NULL. But the efs action will be affecting the traffic based on earlier lookup.

4.13.3 Protobuf structure used for logging:

1. **actionTaken** denotes the final action taken on the packet. Possible values ALLOW or DENY.
2. **malIpAction** denotes Malicious IP engine specific action. Possible values MONITOR or BLOCK
3. **urlCatFilterAction** and **urlRepAction** deontes URL Category and URL Reputation engine specific actions respectively. Possible values are ALLOW or MONITOR or BLOCK.

```
/* By default all the fields are optional in proto3 */
syntax = "proto3";
```

```

enum EfsAction {
    EFS_ACTION_ALLOW = 0;
    EFS_ACTION_MONITOR = 1;
    EFS_ACTION_BLOCK = 2;
}

message MalIpFiltering {
    repeated uint32 ipCategories = 1 [packed=true];
    EfsAction malIpAction = 2;
}

message UrlCatFiltering {
    repeated uint32 urlCategories = 1 [packed=true];
    EfsAction urlCatFilterAction = 2;
}

message UrlRepFiltering {
    uint32 urlReputation = 1;
    EfsAction urlRepAction = 2;
}

message FirewallLogMessage {
    string actionTaken = 1;
    string ruleId = 2;
    uint32 sessionId = 3;
    string segmentLogicalId = 4;
    string inputInterface = 5;
    uint32 protocol = 6;
    string sourceIp = 7;
    string destinationIp = 8;
    uint32 sourcePort = 9;
    uint32 destinationPort = 10;
    string destination = 11;
    string domainName = 12;
    string firewallPolicyName = 13;
    string segmentName = 14;
    string extensionHeader = 15;
    string application = 16;
    uint32 sessionDurationSecs = 17;
    uint32 bytesSent = 18;
    uint32 bytesReceived = 19;
    string closeReason = 20;
    uint32 signatureId = 21;
    string verdict = 22;
    string signature = 23;
    string category = 24;
    uint32 ruleVersion = 25;
    string attackSource = 26;
    string attackTarget = 27;
    uint32 severity = 28;
    string edgeLogicalId = 29;
    string enterpriseLogicalId = 30;
    string edgeName = 31;
    /* Unix epoch time - 64 bits to avoid Y2K38 problem */
    uint64 timestamp = 32;
    /* Set to 1 if it is IDS alert, otherwise 0 */
    uint32 idsAlert = 33;
    /* Set to 1 if it is IPS alert, otherwise 0 */
    uint32 ipsAlert = 34;
    /* url is used in uRLF case */
    string url = 35;
    UrlCatFiltering urlCatFiltering = 36;
    UrlRepFiltering urlRepFiltering = 37;
    MalIpFiltering malIpFiltering = 38;
    repeated uint32 engineType = 39 [packed=true];
    uint32 firewallLogVersion = 40;
}

```

4.13.4 Logging for Malicious IP:

Only when an IP matches is found in Malicious IP DB we will generate log.

Note: If IP is not found in Malicious IP DB it is considered valid IP and we don't generate log.

Below table explains based on config what will be the contents in protobuf fields actionTaken and malIpAction.

actionTaken is the final action happening for the packet. **malIpAction** is engine specific action.

Malicious IP Config	**Final Action (actionTaken)**	**Malicious IP engine specific action (malIpAction)**	
		syslog forwarding / local logging / VCO FW Logging endpoint / VCO Monitoring endpoint	**debug.py --user_firewall_dump**
Allow and Log	Allow	Monitor	Monitor
Block	Deny	Deny	Block

4.13.5 Logging for URL Category and URL Reputation

For URL Category and URL Reputation we will combine the information into one log message.

For URL reputation

- We Block and log everything below min-reputation
- Else if the category is in capture logs.

Below table explains based on config what will be the contents in protobuf fields **actionTaken, urlCatFilterAction and urlRepAction.**

The config columns means that the packet matched the particular config.

URL Category Config	**URL Reputation Config**	**Final Action (actionTaken)**	**syslog forwarding / local logging**		**VCO FW Logging endpoint**		**VCO Monitoring endpoint**		**debug.py --user_firewall_dump**	
			URL Category engine specific action (urlCatFilterAction)	**URL Reputation engine specific action (urlRepAction)**	**URL Category engine specific action (urlCatFilterAction)**	**URL Reputation engine specific action (urlRepAction)**	**URL Category engine specific action (urlCatFilterAction)**	**URL Reputation engine specific action (urlRepAction)**	**URL Category engine specific action (urlCatFilterAction)**	**URL Reputation engine specific action (urlRepAction)**
Block	Block	Deny	Deny	Deny	Deny	Deny	Deny	Deny	Block	Block
Block	Allow	Deny	Deny	N/A	Deny	Not packed	Deny	Allow	Block	Allow
Block	Allow and Log	Deny	Deny	Monitor	Deny	Monitor	Deny	Monitor	Block	Monitor
Allow	Block	Deny	N/A	Deny	Not packed	Deny	Allow	Deny	Allow	Block
Allow and Log	Block	Deny	Monitor	Deny	Monitor	Deny	Monitor	Deny	Monitor	Block

URL Category Config	URL Reputation Config	Final Action (actionTaken)	syslog forwarding / local logging		VCO FW Logging endpoint		VCO Monitoring endpoint		debug.py --user_firewall_dump	
			URL Category engine specific action (urlCatFilterAction)	URL Reputation engine specific action (urlRepAction)	URL Category engine specific action (urlCatFilterAction)	URL Reputation engine specific action (urlRepAction)	URL Category engine specific action (urlCatFilterAction)	URL Reputation engine specific action (urlRepAction)	URL Category engine specific action (urlCatFilterAction)	URL Reputation engine specific action (urlRepAction)
Allow	Allow and Log	Allow	N/A	Monitor	Not packed	Monitor	Allow	Monitor	Allow	Monitor
Allow and Log	Allow	Allow	Monitor	N/A	Monitor	Not packed	Monitor	Allow	Monitor	Allow
Allow and Log	Allow and Log	Allow	Monitor	Monitor	Monitor	Monitor	Monitor	Monitor	Monitor	Monitor
Allow	Allow	Allow	Won't generate log	Won't generate log	Not packed	Not packed	Allow	Allow	Allow	Allow

4.14 Ambiguity in URL Category match

There is a possibility that a flow can be classified to more than one category and each of these category may be spread across list of allowed and denied categories.

Action will be taken with preference to denied categories.

4.15 Processing config blob

Blob Overview:

For every rule IP Reputation can be turned on/off or in monitor mode.

```
"segments": [
   {
    "segment": {
     "segmentId": 0,
     "name": "Global Segment",
     "type": "REGULAR",
     "segmentLogicalId": "45e6ad6f-7613-40a4-8aca-1fce869cf345"
    },
    "firewall_logging_enabled": false,
    "outbound": [
     {
      "name": "AllowAny",
      "match": {
       "ipVersion": "IPv4v6",
       "sipV6": "any",
       "dipV6": "any",
       "appid": -1,
       "classid": -1,
       "dscp": -1,
       "sip": "any",
       "sport_high": -1,
       "sport_low": -1,
       "ssm": "any",
       "svlan": -1,
       "os_version": -1,
       "hostname": "",
       "dip": "any",
       "dport_low": -1,
       "dport_high": -1,
       "dsm": "any",
       "dvlan": -1,
       "proto": -1
      },
      "action": {
       "allow_or_deny": "allow"
      },
      "atp_action" : {
        "atp_enabled": True,
        "ids_enabled": True,
        "ips_enabled": True,
        "atp_logging_enabled": True
      },
+     "urlCategoryFiltering" : {
+        "allowAndLogCategories": [], //list of category IDs
+        "blockedCategories": [], //list of category IDs
+        "unknownCategoryAction" : "allow/block"
+     },
+     "urlReputationFiltering" : {
+        "minReputationScore" : 0-100,
+        "monitorMode" : true/false ,
+        "unknownCategoryAction" : "allow/block"
+     },
+     "maliciousIpFiltering" : {
+         "action" : "allowAndLog/Block"
+     },
+     "ruleLogicalId": "BSouJbfzS7OjyOk0MyaMHA"
    }
   ]
  },
```

4.16 Threat Modeling

This section should minimally address the following

- [ ] **Adding link to feature threat modeling document (2020-03):** Threat Modeling[12]
    - [ ] **Please review the threat model with security by following** Feature Level Threat Modeling[13]**. This will guide you to open a Ticket with the SecOps team.**
- [ ] **Are any new or upgraded components or dependencies involved in the implementation of this feature?**
    - [ ] **If so, please list the components/dependencies and their versions.**
    - [ ] **If electing to use an 'old' version, please explain the use along with the maintenance plan for the new component.**
- [ ] **Does this feature have security implications that needs an infosec review? Default: Yes**
- [ ] **Is customer data protected appropriately?**
- [ ] **Are there any PII or password data that this feature is handling and is the protected appropriately?**
- [ ] **If there are APIs, are the input data sanitized as well?**
- [ ] **If there are UI elements, are they protected from XSS vulnerabilities, etc.?**
- [ ] **Is there a Threat Model for this feature/interface? See above link to "Threat Modeling"**
    - [ ] **Include diagrams which help detail the change to the system, service or component**
    - [ ] **Please consider the impact to our system and customers when bad actors mis-use your feature**
- [ ] **What kind of security testing should be done?**

Threat Modelling for DP and MP are documented as a separate confluence page.

URL Filtering, Malicious IP Filtering - Threat Modeling[14]

4.17 Thread Design Constraints

The edged process comprises of several threads running on different priorities (SCHED_OTHER, SCHED_RT etc.,). This comprises of both DP, CP threads and certain book keeping (sweep timer based) threads which is a mix of critical and non-critical tasks. A mixup of thread priorities can leads to issues like priority inversions

12 https://onevmw.sharepoint.com/:p:/t/VeloCloud-All/ETnNRJaCIHNKhC6HBNyxkhoB0C1JoAjseQjq1B6Q6CJ0tA?e=ifN9Or
13 https://confluence.eng.vmware.com/display/VELOSEC/Feature+Level+Threat+Modeling
14 https://vmw-confluence.broadcom.net/display/VELOENG/URL+Filtering%2C+Malicious+IP+Filtering+-+Threat+Modeling

and CPU starvation issues as well as cause an imbalance at system level across all processes. In effect, we need some guidelines for a thread to be marked as realtime priority in order to be mindful of all other existing threads and processes in the system.

Following are the guidelines for Realtime Thread priority,

- [ ] **Is the task accomplished by the thread critical in nature (DP task, packet forwarding etc.,) ?**
- [ ] **Does the thread perform core functional tasks like scheduling, DMPO, HA ?**
    - [ ] **DMPO**
    - [ ] **HA**
    - [ ] **Scheduler**
- [ ] **Does the thread perform time bound activities and capable of relinquishing CPU time ?**
- [ ] **Does the thread have no locking dependencies (lock ordering etc.,) across different services (CP & DP) ?**
- [ ] **Is the thread void of any spawning of shells and system calls that block the progress of thread ?**
- [ ] **Is the thread void of heavy debug logging operations ?**

Existing thread impacted:

- URLF/IP reputation Config shall be done using config rpc thread
- Lookup shall be done in the exception path.

New threads:

- WR Background thread: will be added by webroot sdk for background activities like network querying, update querying and database updation. Current we are going with 2 threads. Shall revisit upon performance tuning. These threads are spawned with SCHED_RR priority as they operate on shared data using lock with other real time threads.

5 Low level design

This section should address the following

- [ ] **Describe the design of this feature as it applies to the SEBU ecosystem.**
- [ ] **Are there any caveats or limitations? What were the alternative approaches considered? What are the tradeoffs in the selected approach?**
    - [ ] **Detail tabular representation of Design, Performance and Scale limitations for selected and alternative approach if any.**
- [ ] **Explain with sequence & interaction diagrams, if possible.**
- [ ] **Describe all components & modules that will be touched by this feature and explicitly call out the before and after behaviour of those components in relation to this feature.**
- [ ] **Are there new modules being introduced? Explain the reasoning for this module.**

VCO

- [ ] **Are there any special considerations for the on-prem version of VCO?**
- [ ] **Are there any failure modes for the implementation like degraded operation etc.?**
- [ ] **List out the dependencies on the Data/Control plane.**
- [ ] **Indicate the database schema design if it applies.**
- [ ] **Should the newly-added data model be considered for Enterprise Cloning feature and Customer Migration tool?**
- [ ] **Is there a scope of role customization based on requirements in PRD? If yes, call out the privileges, UI fields to hide and explain in detail about the standard roles to be granted/denied.
Define privilege labels and description in i18n file.**

DP

- [ ] **Type your task here, using "@" to assign to a user and "//" to select a due date**
- [ ] **List out the dependencies on the Management Plane including UI/UX changes if needed.**
- [ ] **List out the new data structures and the modification in the existing ones.**
- [ ] **List out Global & Static variables used and in which scenario along with counters & lock used**
- [ ] **Include details on functions/API at component level and the thread context in which these functions are executed and details on thread synchronization.**

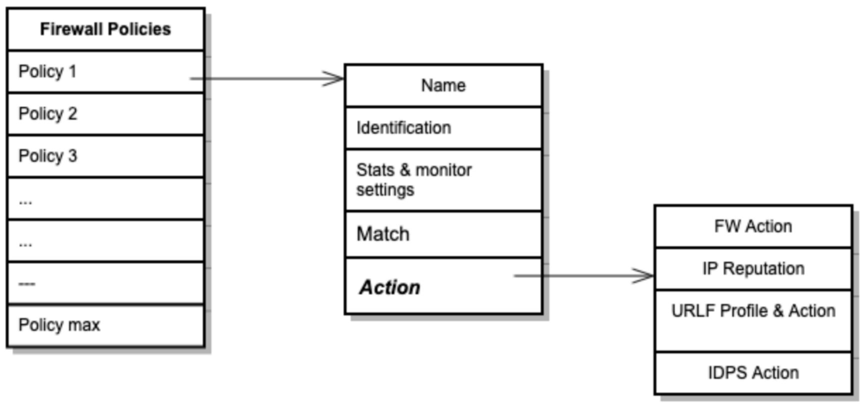
5.1 Dataplane

5.1.1 Data Structure Changes

5.1.1.1 **Firewall policy**

Firewall policy_match and match key doesn't have any change. The same set of parameters in match criteria part is retained.

Firewall Action data structure shall be provisioned to carry more than one action



Firewall policy

```
struct firewall_policy {
    char name[MAX_NAME_LEN];
    char rule_id[MAX_NAME_LEN]; /* Used by VCO for mapping firewall log messages to
the rule in it's own database */
    uint64_t num_hits;
    uint64_t last_log_ts; /* in ms */
    int32_t logging_enabled;
-   uint8_t atp_enabled:1;
-   uint8_t atp_logging_enabled:1;
+   uint8_t efs_enabled:1;
    struct policy_match match;

-   union {
-       struct inbound_fw_action inbound;
-       struct outbound_fw_action outbound;
-       atp_firewall_action_t atp_action;
-   } action;

+   union {
+       struct inbound_fw_action inbound;
+       struct outbound_fw_action outbound;
+       struct efs_profile *efs_object; <============ Ref to security profile
+   } action;
    AO_t refs;
};
```

5.1.1.2 EFS Profile

```
struct efs_profile {
    struct idps_profile idps_obj;
    enum filter_mode mal_ip_mode;
    struct web_reputation_profile url_rep_filter;
    struct url_cat_profile *url_cat_obj;
};
```

5.1.1.3 IDPS Profile

```
// IDPS Filtering action
struct idps_profile {
     enum filter_mode idps_mode;
     bool log;
};
```

5.1.1.4 WebReputation profile

```
struct web_reputation_profile {
    enum filter_mode web_rep_mode;
    uint8_t minimum_reputation;
    uint8_t allow_unknown;
}
```

5.1.1.5 URL Cat Filter

```
// URL Filtering action
struct url_cat_profile {
    urlf_cat_bitmap_t blocked_cat;
    urlf_cat_bitmap_t monitor_cat;
    bool allow_unknown;
 };
```

5.1.1.6 IP reputation mode

```
enum filter_mode {
    EFS_MONITOR,
    EFS_PROTECT,
    EFS_DISABLED
};
```

5.1.1.7 URL Category bitmap

```
// URL Filtering criteria
typedef struct {
    uint64_t catmap[VC_CAT_BITMAP_ARRAY_SIZE];
} urlf_cat_bitmap_t;
```

5.1.1.8 uFlow Container

Micro Flow container shall store the reputation score and category.

```
struct vc_uflow {
    struct vc_uflow_key key;
    uint32_t users;
    uint32_t active_fc_refs;

    union {
        struct flow_tcp_state               tcp;
        struct flow_icmp_state              icmp;
    } proto_state;
+   struct efs_class *efs_data;
-   struct vc_atp_state atp_state;
};
```

5.1.1.9 EFS data in flow

```
struct efs_class {
    struct vc_atp_state                           atp_state; <=========== carried from
uflow
    uint8_t                                       url_reputation; <====== new field
    uint8_t                                       ip_reputation; <======= new field
    struct  url_cat_profile                       url_category; <======== new field

};
```

5.1.1.10 Global Domain name Hashmap object

```
struct domain_name_map {
    char dom_name[MAX_DOMAIN_CHAR]; //128 bytes
    struct fc_list_node head;       // list of fc waiting to be classified
};

struct fc_list_node {
    struct vc_list node;
    struct flow_container *fc;
};
```

5.1.2 Webroot Interfaces

All cpp calls shall be abstracted with an equivalent C function calls

Underlying CPP API calls

Function	**Arguments**	**Description**
Isis::SetConfiguration	config file	specify config file to load config from
Isis::Initialize	Nil	Initialize the webroot sdk
Isis::UrlLoadExisting	Nil	Load existing URL DB
Isis::LookupUrlFromLocalCache	Req(url and result), callback	Make lookup from local cache
Isis::StatusSuccess	lookup status	check request status is successful
GetUrlCategory(cc.category, desc)	category id, output string	get the URL category string from id

Function	Arguments	Description
Isis::LookupUrlInfo	Req(url and result), callback	make lookup from local
Isis::LookupUrlReputation	Req(url and result), callback	URL reputation lookup - non blocking
GetIpReputationInfo	Req, callback	Ip reputation lookup with all info
GetIpThreatInfo	Req, callback	Ip repuattion lookup just with threat identification
Task::GetStats	Output stat counters	To get the webroot API statisticstics
Isis::Shutdown();	Nil	Uninitialize webroot sdk

5.1.3 Webroot SDK configs

1. **URL update time:** Interval at which URL DB gets updated. We shall be updating once per day.
2. **IP Reputation update time:** Interval at which IP database gets updated. once per day proposed
3. **Background threads:** number of threads to process asynchronous requests and updates
4. **Database location:** Disk location where database file downloaded are stored
5. **Credentials:** Device id, OEM, UID
6. **IsisServer:** Server from where updates and query are targeted
7. **Log level:** Set the logging level

5.1.4 C API layer

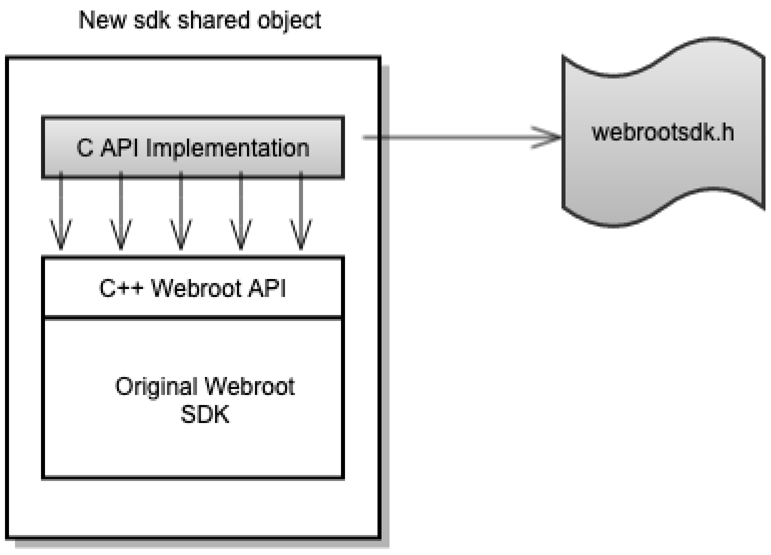
Webroot SDK APIs are written in C++ and hence those APIs follow C++ semantics.

A layer containing C wrapper functions calling those C++ APIs shall be written and compiled along with webroot sdk.

These C functions shall be exported as APIs using a header file (webrootsdk.h).

Those functions shall be linked from libIsisSdk.so[15]

15 http://libIsisSdk.so



5.1.5 Webroot resource locking

Resource sharing critical sections are significant in two scenarios with respect to webroot databases.

1. Database updates
2. Cache update

Database updates are done without locks using rcu (implemented using shared_ptr in cpp).

URL Cache are contended between lookup and network updates. These data are protected by readwrite locks.

Our implementation has frequently accessing more readers than occasionally accessing less writers.

5.1.6 Overloading Log function and new/delete operator

We shall be registering a callback function to overload new operator with vc_malloc and delete with free.

Subsequent calls inside sdk will use vc_malloc to allocate memory and vc_free to free the memory.

Also log function inside Webroot SDK library overloaded with DBG_LOG.

5.1.7 URLF Engine

URL lookup is launched in the exception path.

Webroot sdk provides both blocking and non blocking calls to make a lookup on the DB.

Non blocking call takes a callback API.

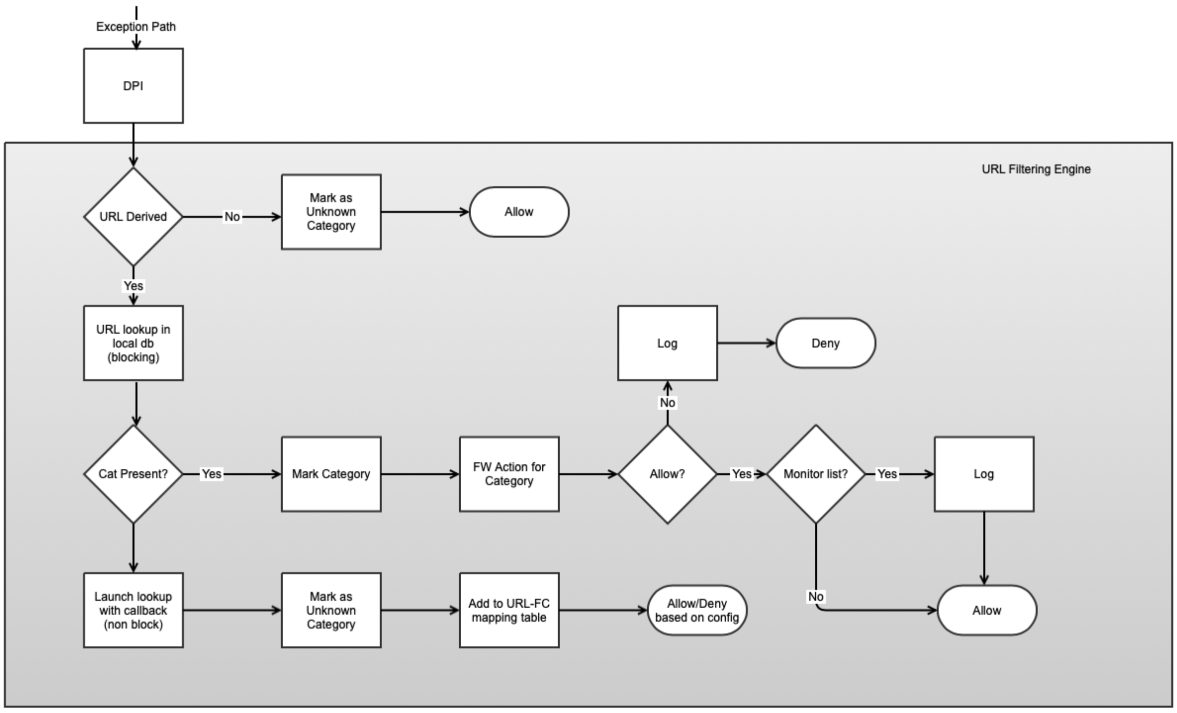
In Order to do inline lookup, initially a blocking call to do a lookup from local cache is launched.

If there is a failure with no_data, a non blocking lookup is launched with a callback and the fc added to a global url-fc mapping table.

This call shall be launching a network query implicitly and the callback is invoked upon reply.

The callback function launches lookup in the url-fc mapping table and update fc versions of all the fc present against the URL/Domian name. This url entry shall be cleared on reply.

Also a thresh-hold timer of 5 Sec is started upon the first fc entry added for a url in the table. When timer expires for the URL entry, each fc in the URL entry is marked as uncategorized & suspicious reputation and fc version gets updated to force the next packet into exception path.

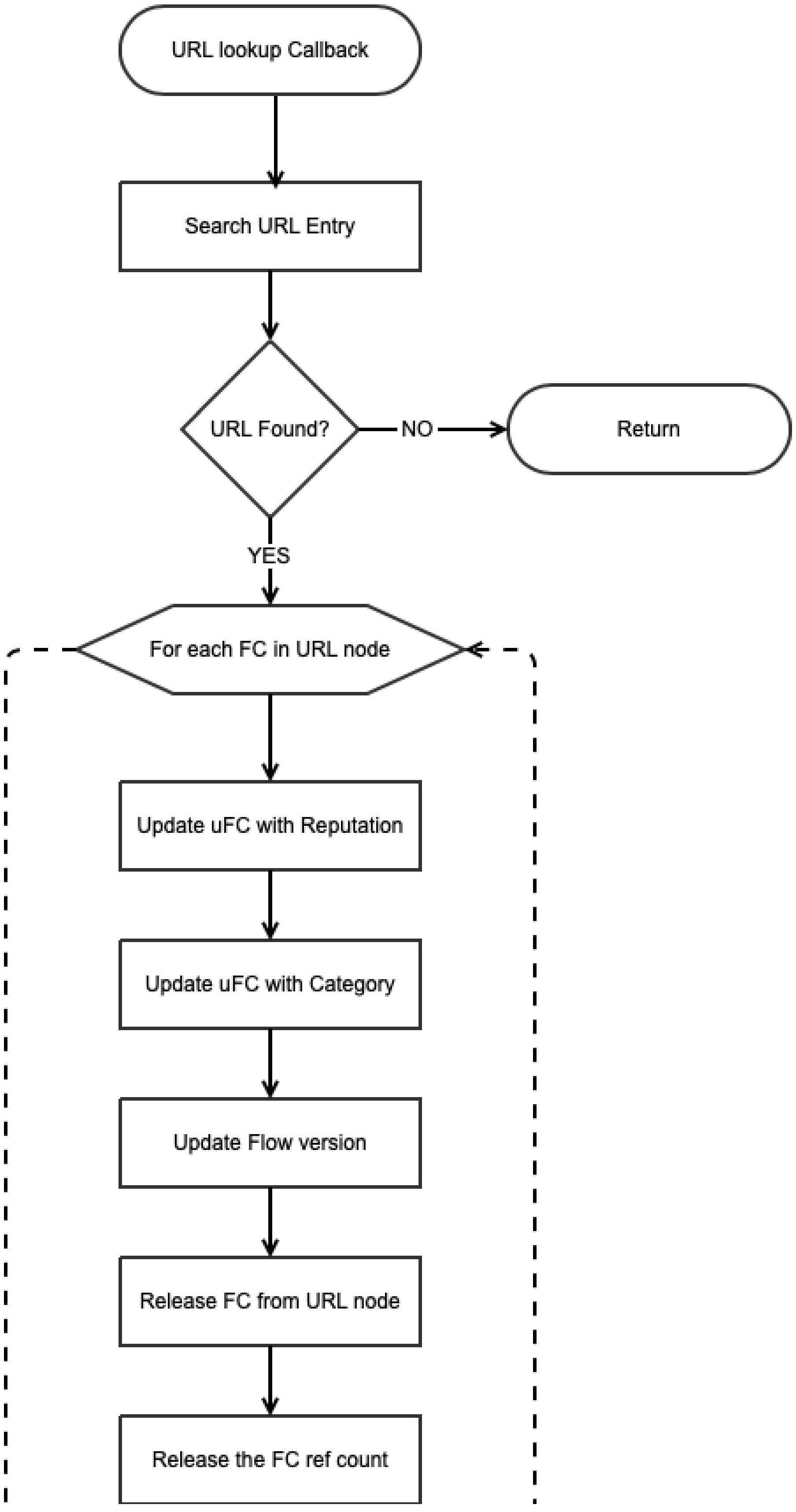


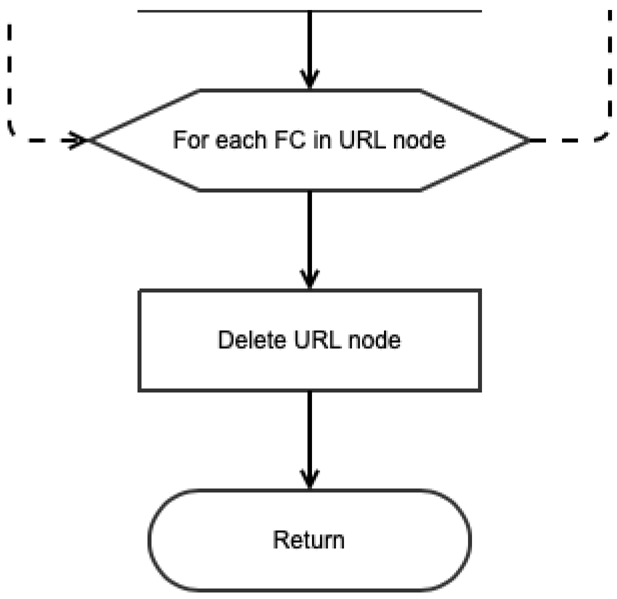
For URL Reputation same flow is applicable except for marking Unknown Category replaced with assigning default reputation value which would allow the packet.

5.1.8 Callback on Network Query

The Callback function is invoked the would return the category and Reputation.

URL is searched upon in the URL-FC mapping db and all the fc(uFC) mapped to the URL are updated with category and reputation. The FC is pruned from the mapping list. Once all PCs are updated, the URL entry is deleted from mapping table.


URL lookup Callback
Search URL Entry
URL Found?
NO
Return
YES
For each FC in URL node
Update uFC with Reputation
Update uFC with Category
Update Flow version
Release FC from URL node
Release the FC ref count



5.1.9 QOSMOS interfaces

Qosmos sdk is used to extract domain name from HTTP and HTTPs traffic.

The following QOSMOS APIs/Data Structures can be used to extract the domain name.

- Registering for attributes for extraction
  The signature/attribute pair can be registered by calling

```
int qmdpi_bundle_attr_register(struct qmdpi_bundle *bndl, char const *sig, char
 const *attr);
```

  This can be done at init time as well as at runtime.

- Then the domain name (SSL_COMMON_NAME, HTTP_URL and HTTP_USER_AGENT) can be extracted from the HTTP/HTTPS packets though

```
void edged_analytics_extract_web(struct session *sess, struct app_desc *adesc)
```

- The attribute SSL_COMMON_NAME is for extracting the domain name from HTTPS traffic
  The attribute HTTP_USER_AGENT is for extracting the domain name from HTTP traffic
  The attribute HTTP_URI_RAW is for extracting the page/file from the HTTP web address
  The default value of uri_raw is '' and so len is 1.
  If len is > 1, concatenate dst_host (domain name) and uri_raw (page) to form the HTTP-URL to query with webroot to decide on allow/deny a page
  In case of HTTPS, uri_raw is (null) and len is 0. So, there is no impact.

  For Eg,

Web Address (HTTP / HTTPS)	dst_host (HTTP_USER_AGENT) (domain name)	uri_raw (HTTP_URI_RAW) (page/file,...)	uri_raw length	ssl_common_name (HTTPS : SSL_COMMON_NAME)	URL passed to webroot SDK dst_host+uri_raw
http://www.example.com	www.example.com[16]	/	1	(null)	www.example.com[17]
http://www.example.com/sports	www.example.com[18]	/sports	7	(null)	www.example.com/sports[19]
https://www.example.com	www.example.com[20]	(null)	0	www.example.com[21]	www.example.com[22]
https://www.example.com/sports	www.example.com[23]	(null)	0	www.example.com[24]	www.example.com[25]

5.1.10 Behaviour for Unknown category/Uncategorized

Unknown:

When we do not have result on local query, the flow is classified as unknown.

Lookup on subsequent packets may succeed because of cache updation by network query.

These flows are allowed/denied based on config momentarily till it gets classified. By default is allowed..

Uncategorized:

Network query for a flow may fetch the result as uncategorized.

Also if the network query doesn't succeed, flows that are classified as unknown shall be marked as Uncategorized after a threshold time of 5 Seconds.

These traffic by default are treated as permit/allow.

16 http://www.example.com
17 http://www.example.com
18 http://www.example.com
19 http://www.example.com/sports
20 http://www.example.com
21 http://www.example.com
22 http://www.example.com
23 http://www.example.com
24 http://www.example.com
25 http://www.example.com

This category shall be configured to drop/allow/reject

Note: network query can yield a URL category as Uncategorized with URL reputation falling in any range.

Unavailable Web Reputation:

Until reputation is available, traffic is allowed.

5.1.11 Edged Behaviour for EFS for packet handling

Please refer section 4.1.2 for more about packet processing order for EFS engine.

5.1.11.1 Edged behaviour for few notable config and events

1. If EFS engine says action as blocked, we drop the packet and don't do further processing.
    a. Blocked traffics are always logged.
    b. Only incase of URLF Category or URLF Reputation engine blocking the packet we will be sending reset and flush the flow.
2. If IDPS engine is enabled alone or along with any EFS engine, **edged** restarts. Post restart, all flows will have "**efs_class**" allocated as part of flow creation.
    a. Note:"**efs_class**"is required to cache info from all the EFS engines within the flows.
3. If IDPS engine is disabled alone or along with any EFS engine, **edged** restarts. Post restart:
    a. "**efs_class**" will not be allocated as part of flow creation.
    b. Based on firewall policy lookup, shall be dynamically allocated if the matched policy has URLF or Mal IP enabled.
4. When either URLF engine or Malicious IP engine is enabled we do Webroot SDK init.
5. When we enable URLF global toggle:
    a. we reset and flush existing web-traffic flows where **efs_class** is either null or **efs_class** is present with **URL_LOOKUP_DONE** not set, so that new flows are subject to DPI for URL/ Domain extraction.
    b. We are checking URL_LOOKUP_DONE because assume a case below:
        i. Enable (URLF) = Decision is made to allow flow we will allow traffic
        ii. Disable (URLF) = Since URLF is disabled URLF EFS decision will not be accounted but will be stored in the flow
        iii. Enable (URLF) = Previous cached category and reputation will be used instead of doing Webroot re-lookup. Decision of traffic flow will be determined by latest URLF category list and URL reputation of the matched firewall policy.
6. When Malicious IP is enabled, no need to rst/flush the flows.
7. Webroot de-inits are done to ensure edges do not unnecessarily communicate to NTICS which is hosted on AWS Lambda.

Config or Event	Edged restart	Reset and flush flows	webroot SDK	Remarks
URL Filtering enabled	No	Yes	webroot inited if it is not inited already	If URLF is enabled then reset and flush flows. If Malicious IP engine is already enabled and then URLF is enabled we will still reset and flush flows.
URL Filtering disabled (if it is the only efs engine enabled)	No	No	webroot de-inited	EFS engine is disabled. So Firewall action will take effect.
URL Filtering disabled (if there is another efs engine still enabled)	No	No	webroot de-inited	EFS engine is not disabled and if we didn't reload then Malicious IP engine is enabled. URLF action for existing flows will be discarded. Only action from Malicious IP will be honoured.

Config or Event	Edged restart	Reset and flush flows	webroot SDK	Remarks
NTICS server updated	No	Yes	webroot deinit and webroot init	This is less common scenario. There is no option to update NTICS server on the fly. Following will be the action taken: MGD receives new NTICS server IP. MGD will get access token for new NTICS server IP. MGD will notify edged about updated NTICS server IP and new access token. Edged will bring deinit webroot. Edged will update the NTICS server configuration and new access token information.

Config or Event	Edged restart	Reset and flush flows	webroot SDK	Remarks
				Edged will reinit webroot. Key things to note: a. edged will not go for restart. Services which are not dependent on webroot or URL Filtering will continue. b. From the step 4 to 6, we will should have at least one of URL Filtering or Malicious IP enabled. Since the security feature is enabled all the traffic matching the firewall policy will be dropped till webroot is reinited and databases are loaded. NSX doesn't have option as of now to update NTICS end point. They will handle when NTICS services are moved to GCP. VCO level event tracking is needed sent to https://status.sase.vmware.com/
Auth token for NTICS connection updated	No	No	This case will come only if webroot is already inited, so just invoke API to update the token	

Config or Event	Edged restart	Reset and flush flows	webroot SDK	Remarks
Malicious IP filtering enabled	No	No	webroot inited if it is not inited already	If Malicious IP filtering is the only efs engine enabled then don't reset and flush flows. If URLF was already enabled and Malicious IP is enabled we don't need to flush flows. Because firewall version mismatch will send subsequent packets to exception path will subject the packets to Malicious IP lookup.

5.1.11.2 **Config Flow-1:**

Security feature from VCO can be enabled only if one of the EFS engine is also enabled. In below example configuration flow we are taking example of

1. Enable(Security Feature + IDPS) → 2. Enable(URL Filtering) → 3. Enable (Malicious IP) → 4. Disable (IDPS) → 5.1 Disable (URL Filtering) → 6. Disable (Malicious IP)

→ 5.2
Disable (Malicious IP) → 6. Disable (URL Filtering)

ConfigStep	Securityfeature	IDPS	URLFiltering	MaliciousIP	Edged EFS and per engine state	Restart required	Reset and flush flows	Webroot SDK	Suricata Engine	Edged behaviour	Edged order of action
Initial	Disabled	Disabled	Disabled	Disabled	EFS = Disabled	N/A	N/A	Not Initialized	Not Initialized		
1.	Enabled	Enabled	Disabled	Disabled	EFS = Enabled IDPS = Enabled	yes	N/A due to restart	Not Initialized	Initialized post restart	Packets will be subjected to IDPS engine alone	IDPS action allow until Suricata engine matches a Deny signature

ConfigStep	**Securityfeature**	**IDPS**	**URLFiltering**	**MaliciousIP**	**Edged EFS and per engine state**	**Restart required**	**Reset and flush flows**	**Webroot SDK**	**Suricata Engine**	**Edged behaviour**	**Edged order of action**
2.	Enabled	Enabled	Enabled	Disabled	EFS = Enabled IDPS = Enabled URLF = Enabled	No(as **IDPS** was enabled in step 1)	yes (as URL was not extracted for existing flows. This will force client to reinitiate flows for webtraffic that will be subject to DPI for URL extraction)	Will init if authentication is successful with NTICS endpoint and edged gets the token	Already up after step 1.	Packets will be subjected to URL Filtering and IDPS engine In URL Filtering it can be any one of the following 1. URL Category filtering 2. URL Rep filtering 3. URL Category and URL Rep filtering	IDPS action allow until Suricata engine marks as Drop Once URL is extracted by DPI packet is subjected to EFS engine in below order 1. url reputation filtering if configured 2. url category filtering if configured

ConfigStep	Securityfeature	IDPS	URLFiltering	MaliciousIP	Edged EFS and per engine state	Restart required	Reset and flush flows	Webroot SDK	Suricata Engine	Edged behaviour	Edged order of action
3.	Enabled	Enabled	Enabled	Enabled	EFS = Enabled IDPS = Enabled URLF = Enabled Mal IP = Enabled	No(as **IDPS** was enabled in step 1)	no (For malicious IP lookup the key is just the destination IP. So even in a middle of ongoing flows, we should be able to enable and subject it to the engine)	SDK was already initialized after step 2	Already up after step 1.	Packets will be subjected to IDPS engine , Malicious IP and URL Filtering. In URL Filtering it can be any one of the following 1. URL Category filtering 2. URL Rep filtering 3. URL Category and URL Rep filtering	IDPS action allow until Suricata engine marks as Drop First packet is subjected to Malicious IP engine and action is taken. If malicious ip engine = allow and once URL is extracted by DPI packet is subjected to EFS engine in below order 1. url reputation filtering if configured

ConfigStep	**Securityfeature**	**IDPS**	**URLFiltering**	**MaliciousIP**	**Edged EFS and per engine state**	**Restart required**	**Reset and flush flows**	**Webroot SDK**	**Suricata Engine**	**Edged behaviour**	**Edged order of action**
											2. url category filtering if configured

ConfigStep	Securityfeature	IDPS	URLFiltering	MaliciousIP	Edged EFS and per engine state	Restart required	Reset and flush flows	Webroot SDK	Suricata Engine	Edged behaviour	Edged order of action
4.	Enabled	Disabled	Enabled	Enabled	EFS = Enabled URLF = Enabled Mal IP = Enabled	yes, as IDPS has been disabled	N/A due to restart	Will be re-initialized post restart	Will not be initialized post restart	Packets will be subjected to Malicious IP and URL Filtering. In URL Filtering it can be any one of the following 1. URL Category filtering 2. URL Rep filtering 3. URL Category and URL Rep filtering	First packet is subjected to Malicious IP engine and action is taken. If malicious ip engine = allow and once URL is extracted by DPI packet is subjected to EFS engine in below order 1. url reputation filtering if configured 2. url category filtering if configured

ConfigStep	**Securityfeature**	**IDPS**	**URLFiltering**	**MaliciousIP**	**Edged EFS and per engine state**	**Restart required**	**Reset and flush flows**	**Webroot SDK**	**Suricata Engine**	**Edged behaviour**	**Edged order of action**
5.1	Enabled	Disabled	Disabled	Enabled	EFS = Enabled Mal IP = Enabled	No (as IDPS was disabled in step 5)		SDK was already initialized after step 2	N/A	Packets will be subject to Malicious IP alone.	First packet is subjected to Malicious IP engine and action is taken.
5.2	Enabled	Disabled	Enabled	Disabled	EFS = Enabled URLF = Enabled	No (as IDPS was disabled in step 5)		SDK was already initialized after step 2	N/A	once URL is extracted by DPI packet is subject to EFS URL Filtering engine	Once URL is extracted by DPI packet is subjected to EFS engine in below order 1. url reputation filtering if configured 2. url category filtering if configured

Config Step	Security feature	IDPS	URL Filtering	Malicious IP	Edged EFS and per engine state	Restart required	Reset and flush flows	Webroot SDK	Suricata Engine	Edged behaviour	Edged order of action
6.	Disabled	Disabled	Disabled	Disabled	EFS = Disabled	No (as IDPS was disabled in step 5)		Webroot SDk deinit	N/A	Only Standard/ legacy firewall action is applicable	Edged does firewall policy lookup and carries out the action in the policy.

5.1.12 NTICS Registration

License key and end point information are obtained from the VCO HB.

When license key is received, registration process begins with the NTICS. Upon registration we get client id and client secret for further communication. This is usually a onetime process per system and in case of HA systems, if client id and secret is not synced to standby and switchover happens, we need to register again.

When a license key is received and if it's different from old license key, re-registration happens and new clientid and secret are maintained and synchronised to standby. In this case, the current token is invalidated.

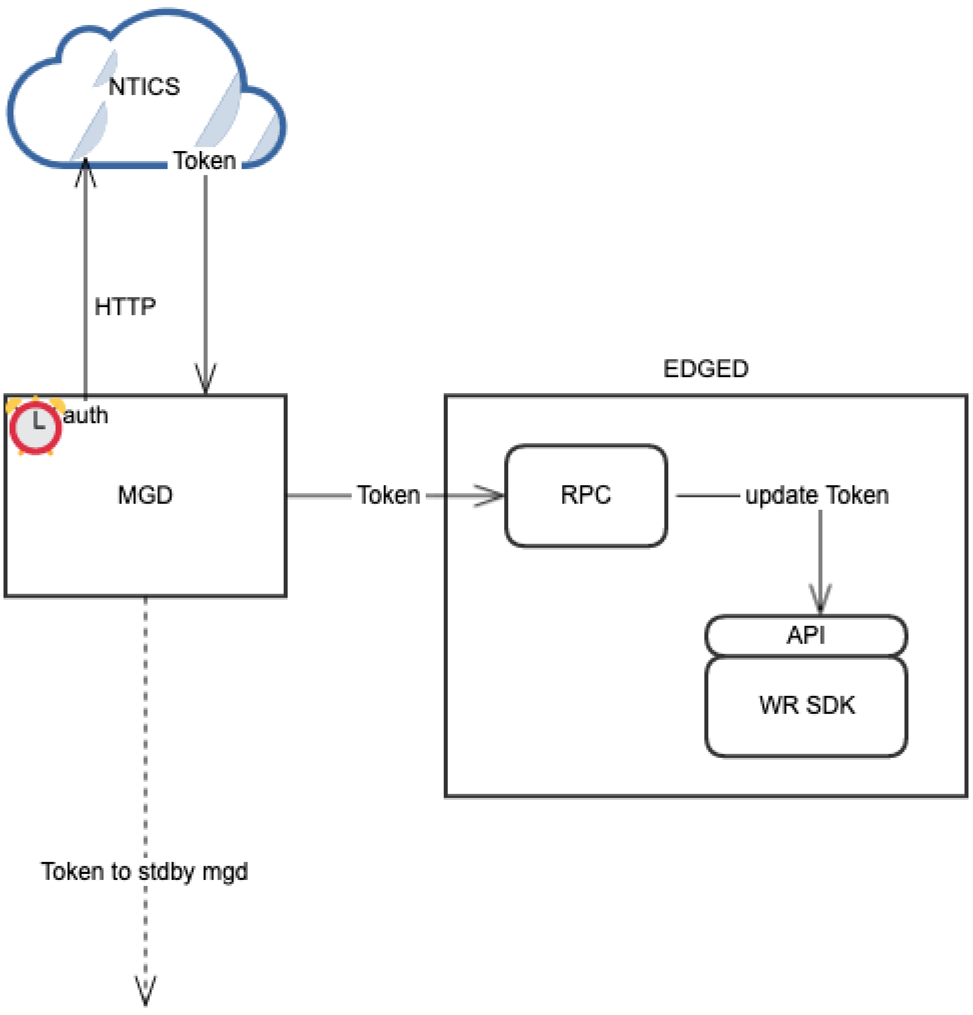
Whenever a registration is successful or unsuccessful, an Edge event is generated towards VCO with client-id(null client id in case of unsuccessful registration).

5.1.13 Auth Token update

MGD process will be periodically authenticating with the NTICS server and the token shall be updated to edged using RPC interface.

A new interface in webroot shall be implemented to update the auth token. This shall be used to update the token that should used for any HTTP communication from webroot sdk to NTICS.

A separate api to update the NTICS end point information into webroot sdk for communication will also be implemented to update url for communication.



MGD token authentication happens on the following events

1. Token validity expires
2. Mgd reloads and new config is read
3. whenever URLF/IP reputation enabled

MGD token sync happens on following events

1. To edged, when authentication succeeds
2. To standby, when authentication succeeds
3. To standby, when new standby is seen
4. To standby, when split brain resolved

5. To edged, when received from active
6. To edged, after edged reloads

5.1.14 Callback registration for Event reporting

In order to report events such as DB version update, RTU updates, IP/URL DB refresh, etc., we will be registering callbacks with webroot SDK. Whenever callback is hit, appropriate events shall be sent out. MGD is responsible to do reauthentication before token expiry. In some corner case, where it missed to perform reauth, and when webroot uses that in the NTICS query, it shall get a invalid token response. Since we register callbacks with webroot, we can use it to trigger mgd to perform re-authentication.

Summary of callback purposes

- Log Events
- Auth failure events
- DB update Events

5.1.15 File locations

These are the proposed absolute path used by webroot sdk

1. Database - **/velocloud/webroot/*database*.bin**
2. Config - **/etc/bcti.cfg**
3. logs - **/var/log/bcti.log**

The log file shall be added to log archiver list so that old logs are archived under /velocloud.

5.2 API, Events & System Properties

This section should minimally address the following

API

- ☐ **Does the feature impact the existing API schema? e.g. Does it call for a modification to device configuration data? If so, has the feature undergone mandatory API review by PM & API team?**
- ☐ **Are there new APIs being introduced? Are these public? If so, has the feature undergone mandatory API review by PM & API team?**
- ☐ **Is there any backward compatibility impacting changes that should be tested, with respect to APIs and functional behavior?**

APIv2

- ☐ **Are there any changes to schema that is not reflected in APIv2 schema? Are they reviewed by the API team?**
- ☐ **Are there any changes to existing APIs that could affect APIv2 behavior as well? For example, addition of filtering operators, fields for monitoring APIs**
- ☐ **Is there any new API introduced which should also/only be part of APIv2 going forward?**

System Properties

- ☐ **Are new system properties being introduced? Explain their defaults and scenarios under which they should be updated, with example values.**

Events

- ☐ **Are new "EVENTS" being introduced as part of this feature? Please list the following details for "each" new event introduced:**
    - ☐ **EVENT NAME: unique name defined for the event for eg. MGD_ACTIVATION_SUCCESS**
    - ☐ **NOTIFICATION CONFIGURABLE: Is it an event for each an email/sms notification can be configured on the VCO?**
    - ☐ **GENERATED BY: Which component within the VCO/VCE is responsible for generating this event. for eg. edged, mgd, procmon, vco-upload, vco-backend, etc. When generated by VCE, there must be a readable string for the event added on vco code.**
    - ☐ **GENERATED WHEN: Conditions/circumstances under which this EVENT is generated**
    - ☐ **FREQUENCY: How often is this event usually generated, if applicable.**

DP

- ☐ **Change in existing debug command and its impact on existing automation cases . Will it be back-ported to previous releases?**
- ☐ **Impact on getpolicy.py output due to new changes**

5.3 Platform & System Dependencies

This section should minimally address the following

- ☐ **Any changes in DPDK? requires any formal platform testing?**
- ☐ **Any impact to MGD?**
- ☐ **Should performance testing be run explicitly for this feature on different hardware?**
- ☐ **Any impact to High Availability?**
- ☐ **Will this feature require a new Manufacturing Release (MR) image? On which devices? See comment below.**

5.3.1 Impact on System Memory

Impact Area	Definite Impact	Optional impact
Flow	Add: 8 Bytes - Pointer to EFS data Del: 8 bytes - Size of struct vc_atp_state (including padding) Net impact: Nil	24 bytes per flow when configured
Library	Add: 20 MB for IP Database Add: 20 MB for URL DB Add: 20 MB for Cache Add: Miscellaneous 10 MB Net impact: 70 MB	N.A
Domain-FC Map	N.A	Per URL → 256 bytes Per Flow → 48 bytes There is periodic age out in 5 sec per URL. Provision for anticipating 10k flows and 1000 URL at any point of time maximum. Net impact - 960 KB (URL) + 128 (KB) = 1088 KB

5.4 Upgrade & Migrations

This section should minimally address the following

- [ ] **Will this feature cause configuration updates to edges during vco upgrade? It should never.**
- [ ] **How will the upgrade be handled for this feature? Specifically, from older releases to the release containing this feature. Also, from the release containing this feature to newer releases.**
- [ ] **Are there any releases that cannot be upgraded from, for this feature?**
- [ ] **Any impact to Interop combinations (Edge, Hub, Gateway being on different versions)?**
- [ ] **Are there any special considerations for the on-prem or hosted version of VCO?**
- [ ] **How long will the upgrade take?**
- [ ] **Are there any special considerations for the migration tool?**
- [ ] **What additional impact is present for the hosted CloudOps team? Like special rollout procedures, manual updates etc.**

Interoperability testing:

- Should interop with all existing supported end-points.
- Config shall not any create impact on unsupported versions
- Upon upgrade to supported version, feature configs shall take effect

5.5 Scale & Performance

This section should minimally address the following

- ☐ **How does this feature affect VCO scale and performance?**
- ☐ **How does this feature affect VCE or VCG scale and performance - route scale, tunnel scale, IMIX performance?**
- ☐ **Are there any special considerations – disk space, additional memory, CPU?**
- ☐ **How does this affect the runtime profile for portal, upload and backend?**
- ☐ **How is the VCO protected?**
- ☐ **Is there a degraded state under which the VCO can operate, if something goes wrong with this feature?**
- ☐ **Can issues with this feature bring down a production VCO?**

5.5.1 Flow scale

We observed that PoC had only around ~70 MB increase in memory after integrating webroot library with 20 MB database.

There would be some impact on low end boxes due to this, we shall derive supported numbers during testing phase.

5.5.2 Performance impact

It is estimated from PoC that each Webroot URL lookup through sdk could cost around 150 microseconds.

And this lookup is done only once per flow. So we can assume the following.

- CPS could be marginally impacted
- Throughput impact is estimated to be negligible.

5.5.3 Scale numbers

The memory impact of this feature may grow marginally when the cache gets expanded. But not expected to impact much after initialization.

Except for Low end models, we expect the scale to be intact with that of existing supported levels.

Though flow scale will be intact, control plane flows will be created per network query when local lookup fails. This is set to affect the data traffic flow count momentarily.

As we cache the category/reputation details within the flow_container, size of flow_container struct is increased by 18 bytes (1112 + 18 = 1130 bytes). If we consider the below scenario on a low-end platform such as 610:

610	4	8 GB	240k	4.3 MB (18 bytes/flow)

5.6 Operations impact / Supportability

This section should minimally address the following

- ☐ **How does this feature impact TechOps procedures and processes if at all?**
- ☐ **Does TechOps need to modify their existing tooling (ansible/terraform/others) to support this feature?**
- ☐ **How will this feature be monitored once in production?**
- ☐ **How do we know the feature works as intended in production?**
- ☐ **List out the important log messages, performance counters, alerts and events (Details of how to add monitoring metrics for VCO** wiki link[26]**). Note that these are the only artifacts that will be available for troubleshooting VCO/VCE/VCG.**
- ☐ **Are there special debug flags?**
- ☐ **If there are issues in this feature, can this be debugged using the VCO diagnostic bundle?**
- ☐ **There is no CLI as part of our product today. This may be introduced in a later release and will mirror the existing Test & Troubleshoot interface that is exposed via Remote Diagnostics. Any feature that you develop must be supportable without ever accessing the CLI – this means either native VCO visibility under the Monitoring tab or new Remote Diagnostics that explicit cover any troubleshooting required for an end-user. Specifically debug.py is used only by Dev & QA and should never be exposed in the public documentation.**

Events:

The following events are generated sent from VCE towards VCO

MGD_EFS_NTICS_REGISTRATION_SUCCESS	NTICS Registration sucess with client id
MGD_EFS_NTICS_REGISTRATION_FAILURE	NTIC registration failure with retry count
MGD_EFS_NTICS_AUTHENTICATE_SUCCEEDED	NTICS Authentication sucess
MGD_EFS_NTICS_AUTHENTICATE_FAILED	NTICS Authentication failure

26 https://confluence.eng.vmware.com/display/VELOENG/What+Metrics+To+Add+For+Your+VCO+Service

MGD_EFS_URL_DB_VERSION_UPDATE	URL DB Version update
MGD_EFS_IP_DB_VERSION_UPDATE	IP DB version update
EFS_IDPS_NOT_READY	Sent when Suricata engine is not ready and packets are dropped
EFS_URLF_MAL_IP_NOT_READY	Sent when Webroot SDK is not ready and packets are dropped
WEBROOT_NTICS_HTTP_ERROR	When there is HTTP communication error between edged and NTICS
EFS_URL_RTU_DB_VERSION_UPDATE	When RTU update happens for URL DB
EFS_IP_RTU_DB_VERSION_UPDATE	When RTU update happens for IP DB

5.7 Supportability Debug commands

5.7.1 Firewall flow dump

User firewall dump shall include the output with URL cat, URL Rep and IP Rep

edge:b3-edge1:~# debug.py --user_firewall_dump

```
SRC_IP         DEST_IP  SRC_PORT  DEST_PORT  PROTO  DSCP    APP      TCP_STATE  RULE
BYTES_SENT  BYTES_RCVD  DURATION  LAN_SIDE_NAT  NAT_SIP  NAT_SPORT  NAT_DIP
NAT_DPORT  IDPS_ACTION  SIGNATURE_ID  SEVERITY  URL_REP  URL_REP_ACTION URL_CAT
URL_CAT_ACTION   MAL_IP_HIT    MAL_IP_ACTION
169.254.8.2   1.3.0.1       443      39256    TCP     0  https  SERVER_LISTEN   N/A
40.5 KiB    39.0 KiB   1:20:30           N/A      N/A        N/A      N/A         N/A
N/A           N/A       N/A       85    ALLOW     5,8,12.        ALLOW        Y
ALLOW
```

5.7.2 Config dump

debug.py --firewall shall include a new field to show if IP Reputation for the segment is enabled

edge:b3-edge1:~# debug.py --firewall 0

```
Stateful Firewall : Disabled
ATP Firewall Status : Disabled
Firewall logging for segment 0
Global firewall logging  : Disabled
IP Reputation : Enabled <====================================================== Newly
added
====================== FIREWALL ====================
--------------- Firewall Rules ---------------
Name                      Hits  Action
AllowAny                  2438   allow
internal_default_rule        0   allow
velocloud_mgmt_outbound      0   allow
URL_cat_rule                 1   deny
URL_rep_rule                 3   deny

------------- NextGen Firewall Rules -------------
Name   Hits  IDS  IPS

====================== INBOUND INTERNET ACL ====================
-----------------PORT FORWARDING-----------------
Name   Protocol  Interface  Outside IP  WAN Port(s)  LAN IP  LAN Port  Segment Id
Hits

-----------------ONE-TO-ONE NAT------------------
Name   Interface  Outside IP  Inside IP  Bidirectional  Allowed Protocol  Allowed
Port(s)  Segment Id  Hits

====================== INBOUND INTERNET V6 ACL ====================
-----------------PORT FORWARDING-----------------
Name   Protocol  Interface  Outside IP  WAN Port(s)  LAN IP  LAN Port  Segment Id
Hits

-----------------ONE-TO-ONE NAT------------------
Name   Interface  Outside IP  Inside IP  Bidirectional  Allowed Protocol  Allowed
Port(s)  Segment Id  Hits
```

5.7.3 Webroot stats

The following fields of webroot stats are shown debug.py –webroot_stats_dump

debug.py --wrsdk_status

```
edge:b2-edge1:~# debug.py --wrsdk_status
```

```
{
  "ntics_reachable": true,                =====> NTICS server is reachable. Test
connection via webroot works.
  "stats": {
    "cache": 0,
    "cdbapp": 0,
    "cdbconnection": 0,
    "cdbfile": 0,
    "cdbip": 0,
    "cdburl": 0,
    "cloud": 0,
    "dp": 0,
    "filemd5": 0,
    "ipg": 0,
    "ipr": 0,
    "ipt": 10,
    "net": 0,
    "phishingscore": 0,
    "tievidence": 0,
    "tihistory": 0,
    "timd5": 0,
    "tiurl": 0,
    "uc": 0
  },
  "wrsdk_db_loaded": true,              =====> URL DB and IP DB are downloaded and
loaded from NTICS by Webroot SDK
  "wrsdk_init_done": true               =====> Webroot SDK init is done
}
edge:b2-edge1:~#
```

The stats field definition

- cache – number of times data were retrieved from a previous lookup
- cloud – number of times data were retrieved from the cloud
- db – number of times data were retrieved from the local dB
- net – number of times a network lookup occurred, regardless of result
- uc – number of times a URL info task returned an “Uncategorized” result
- ipg – number of successful* IP geo lookups
- ipr – number of successful IP reputation lookups
- ipt – number of successful IP threat lookups
- cdbapp – number of times CDB app stats were successfully read
- cdbconnection – number of successful CDB URL connection lookups
- cdbfile – number of successful CDB file lookups
- cdbpp – number of successful CDB IP lookups
- cdburl – number of successful CDB URL lookups
- filemd5 – number of successful MD5 file lookups
- phishingScore – number of successful phishing score lookups
- timd5 – number of successful lookups of threat insight MD5s
- tiurl – number of successful lookups of threat insight URLs
- tihistory – number of successful lookups of threat insight URL histories

- tievidence – number of successful lookups of threat insight IP evidences

5.7.4 Debug command to induce lookup

We shall be introducing a debug.py command to induce a lookup from edged.

5.7.4.1 Debug command for URL lookup

debug.py --efs_url_lookup

```
Lookup Result: Found in cache/Not found in cache
URL: <url name>
Category id(s): <cat1>, <cat2>, ...
Reputation: <0-100>
```

Example:

```
edge:b2-edge1:~# debug.py --efs_url_lookup www.zoho.com
Lookup Result: local_db
URL: www.zoho.com
Category: Business and Economy, Computer and Internet Info
Reputation: 96
edge:b2-edge1:~#
```

5.7.4.2 Debug command for Malicious IP lookup

debug.py --ip_threat_lookup

```
edge:b2-edge1:~# debug.py --ip_threat_lookup <IPv4>
IP: <IPv4>
Threat Type: <Type1>,<Type2>
Status: local_db
```

Example:

```
edge:b2-edge1:~# debug.py --ip_threat_lookup 91.215.42.31
IP: 91.215.42.31
```

```
Threat Type: Phishing
Status: local_db
```

5.7.5 URL/IP lookup from Remote diag

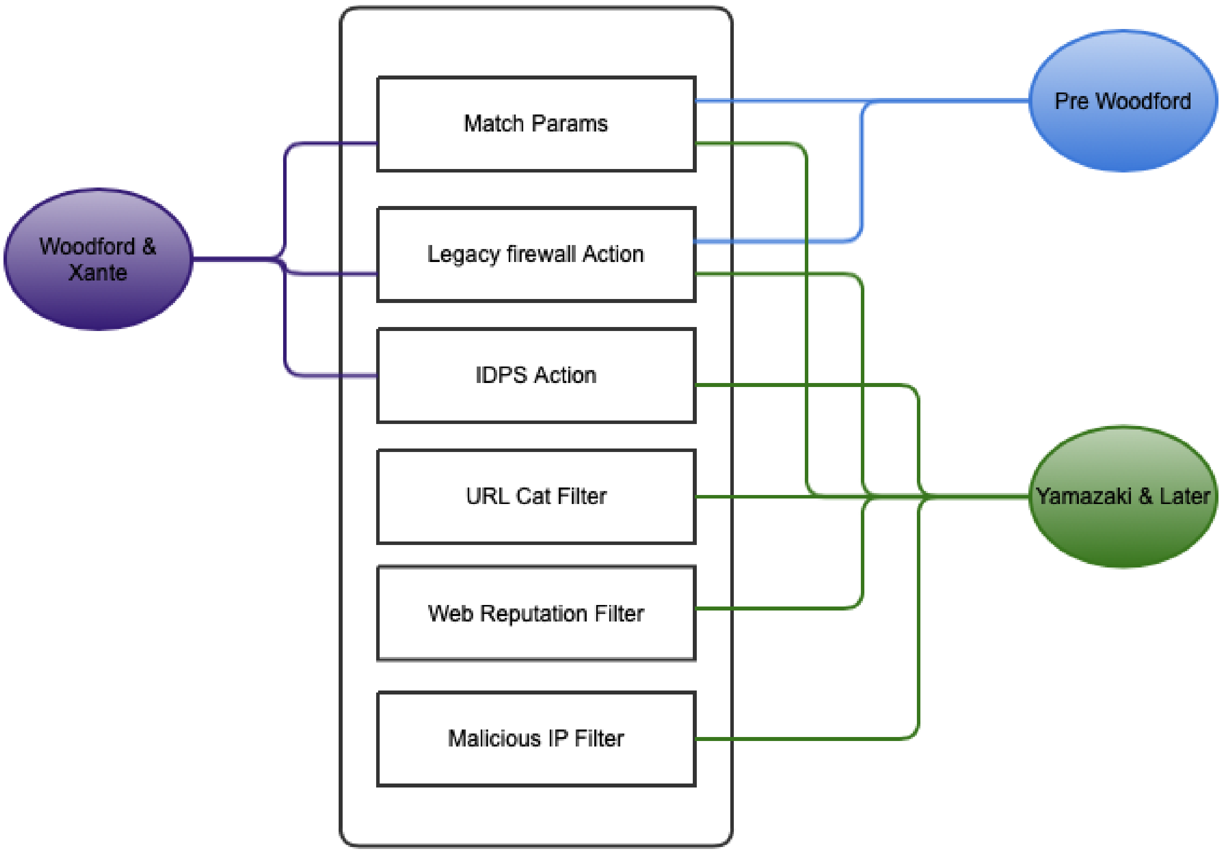
Remote diagnostic page shall facilitate IP/URL lookup to obtain reputation/category for user to debug/evaluate category and reputation classification by the database vendor.

5.8 Backward compatibility

This section should minimally address the following

- [ ] **Does this feature have any backward compatibility breaking changes?**
- [ ] **Is this featured modifying existing data on a production VCO to work (i.e., json/sql patches)**

The JSON structure is designed to be backward compatible. Where legacy firewall action and atp action are sent along with security profile mapping.



Firewall rule is the common config blob between different software versions.

Pre Woodford software, read only the legacy firewall action and infer it as Allow action for the rule.

Woodford and Xante software will read IDPS action and legacy action.

Yamazaki and Later software will read on legacy firewall action, idps action, URL Cat Filter, URL Reputation Filter and Malicious IP filter blobs

Woodford and Pre-woodford, Xante edges will ignore license key configuration. So post upgrade to version above Xante i.e., Yamazaki, the edges will parse the license key and NTICS endpoint configuration from configuration and then do registration/authentication with the NTICS endpoint. VCO shall be adding a patch to obtain license key and updating the EFS related config keys while upgrading Woodford VCOs with EFS enabled.

6 Testing

6.1 General approach

This section should minimally address the following

- [ ] **What is the scope of developer testing?**
- [ ] **For Management Plane, how will the tests be written – Jest/JS regression/Hapy?**
- [ ] **For Data Plane, how will the tests be written – CUnit/Hapy?**

General approach is to test unit by unit as they are developed. The strategy is to develop debug commands to verify along with completion of specific functional or unit change.

For those items where we do not have a debug.py command, we shall be using logs appropriately to verify the behaviour.

Test Items	**Approach**
Configuration	debug.py command
Registration	Logs
Authentication	Logs
Rule match	Logs
HA Switchover & Sync	Logs and command
Memory usage	Command
Database updates	Log

- Longevity and stress tests are covered as part of system testing along with functional triggers.
- Periodically clean up of configs will be done during stress/longevity tests and check if all memory blocks pertaining to these configs are released.

6.2 Acceptance Test

This section should minimally address the following

- [ ] **List of test cases provided by QE to developer and must be passing 100% before QE handoff.**

6.3 Unit test

This section should minimally address the following

- [ ] **What are the unit tests that will be written vs. existing unit tests that cover the new functionality?**
- [ ] **Can you guesstimate the coverage targeted for the tests?**
- [ ] **What has not been unit tested?**
- [ ] **Add TCs to automated Unit test framework and must be passing 100% before QE handoff (code reviewers need to be empowered to unapprove feature branch merges w/o automated UT scripts)**

6.3.1 Manual UT cases

1. Verify if config is pushed from MP to DP
2. Verify if cfg file is packaged as part of upgrade
3. Verify if Ip DB and URL DB is packaged as part of image
4. Verify if firewall rules are configured with different actions for URL Filtering - Cat/Reputation
5. Verify if firewall rules are configured with different actions for IP repuatation.
6. Verify if firewall rules are successfully removed.
7. Verify if firewall rules with URLF/IP rep are configured with any-any-permit in unsupported software versions
8. Verify if Client id and Client secret are received with config from VCO
9. Verify if Mgd authenticate to NTICS cloud using client id and client secret
10. Verify if Mgd, periodically authenticates to obtain auth token from NTICS
11. Verify if edged with the auth token could call various http methods in `X-NCS-Authorization` header field
12. Verify if traffic when URL extracted makes successful lookup in local db
13. Verify if lookup in local db fails, launches a network query.
14. Verify if category is found and firewall rule is matched, action is taken

15. Verify if firewall rule is matched, log is generated
16. Verify if network query is successful, log is generated
17. Verify if flow is uncategorized, it matches the uncategorized firewall rule and appropriate action is taken.
18. Verify if uncategorized traffic are permitted by default if no rule configured
19. Verify if debug.py shows the firewall rule is hit
20. Verify id debug command to make URL lookup work properly
21. Verify if traffic's IP lookup matching Ip database dropped when configured
22. Verify if traffic's IP lookup matching Ip rep database is allowed when not configured.
23. Verify if database file is updated periodically with the given frequency by checking update timestamp.
24. Verify if db file downloads automatically if not present.
25. Verify firewall configs are synchronized between active and standby
26. Verify if flows that firewall actions updated are synchronized to stand by
27. Verify if the flows matching the firewall rules are matched after switchover and new active takes appropriate action.
28. Verify if DNS requests are generating dummy query to URL database
29. Verify if background threads for asynchronous processing are created.

6.3.2 Snitch UT Automation cases

<<TBD>>

6.4 Scale & Performance Test

This section should minimally address the following

- [ ] **What is the customer requirement with respect to scale for this feature?**
- [ ] **Do the scale simulators need to change for this feature?**

- It is expected that for every network query, a flow will be created if not found in cache.
- Performance (CPS) may slightly be impacted due to additional lookups to fetch URL category/ reputation and IP reputation.

6.5 Upgrade / Interoperability Test

This section should minimally address the following

- [ ] **What are the areas (Like Tunnel, Route or MP - Config management) that may get affected during Interoperability?**
- [ ] **Will this feature cause configuration updates to edges during vco upgrade? It should never.**

7 Future considerations & Unsupported items

- [ ] **Anything in this feature that is not covered as part of this FS but will need to be done later**
- [ ] **Any serviceability area for this feature that would have to be done later.**
- [ ] **List of unsupported items.**

8 Documentation Impact

This section should minimally address the following

✓ **If any documentation needs to be added or updated for this feature, please call out those sections here.**

Configuration guide should be updated with the step by step configuration.

Supported URL categories and IP threat types shall be published.

Firewall logs section needs to be updated with additional fields.

9 Functional Specification - User config related enhancements on legacy Firewall (Yamazaki)

Author(s)	DP/CP: @Unknown User (saravanank) MP: @Praveen Kumar Rajendran
Approver(s)	☐ @Aditya Agarwal ☐ @Unknown User (qxinzhou) ☐ @Preethi Nandakumar ☐ @Sathya Thammanur ☐ @Ritesh Tiwari
Reviewer(s)	@Ramprasath G R @Bhuvaneswar Doni @Avinash Bhoomireddy @Preethi Nandakumar @Aditya Agarwal @Karthik Paramasivan <QE Lead, QE Manager, Tech Ops Lead, Security, Product Mgr, Support Engineering, Tech Pubs, UX, API> *Please note that API team review is **mandatory** if the feature impacts the public API (e.g. device configuration schema).
Status	DRAFT
Dev lead/ Architect	DP: @Unknown User (saravanank) MP: @Praveen Kumar Rajendran
DP contact	@Preethi Nandakumar
MP contact	@Preethi Nandakumar
QE contact	@Preethi Nandakumar
Target Release	Yamazaki

PRD	Firewall Policy Enhancements - Woodford and Beyond[27]
Aha Link	
Jira Epic	VLENG-100211[28] VLENG-97388[29]
Project Page	
References	

9.1 Change log

0.1	07 Jul 2023	Initial draft.

9.2 Introduction

9.3 Dependencies

This feature has changes in UI and hence we have MP/UI/Ux dependency

9.4 Risks

This section should minimally address the following

27 https://vmw-confluence.broadcom.net/display/VELOPROD/Firewall+Policy+Enhancements+-+Woodford+and+Beyond
28 https://jira.eng.vmware.com/browse/VLENG-100211
29 https://jira.eng.vmware.com/browse/VLENG-97388

9.5 Functional Overview

9.5.1 5.1. Engineering Requirements

9.5.1.1 5.1.1. **DP Requirments**

9.5.1.2 5.1.2. **MP requirements**

<To be filled by MP>

9.5.1.3 5.1.3. **Supportability Requirements**

9.5.1.3.1 5.1.3.1. DP supportability

9.5.2 5.2. Requirement Traceability Matrix for DP

Require ment	Description	FS Section		Function Test Plan

9.5.3 5.3. High level design

9.6 6. Backward Compatibility

This section should minimally address the following

9.7 7. Security Impact

9.7.1 7.1. Threat modeling

Description:

Data flow diagram:

Input Data validation:

Customer/System impact:

Attacker mis-use scenarios:

9.8 8. Platform & System Dependencies

This section should minimally address the following

9.9 9. API, Events & System Properties

This section should minimally address the following

9.10 10. Upgrade & Migrations

This section should minimally address the following

9.11 11. Operations Impact/Supportability

9.12 12. Scale Impact

9.13 13. Detailed Design & Implementation

9.13.1 13.1. Data Plane

9.13.1.1 13.1.1. Datastructure Changes

9.14 14. Testing

9.14.1 14.1. Unit Testing

9.14.2 14.2. System Testing

This section should minimally address the following

9.14.3 14.3. Scale Testing

This section should minimally address the following

9.14.4 14.4. Upgrade / Interoperability Testing

This section should minimally address the following

Interoperability testing:

9.14.5 14.5. Documentation Impact

This section should minimally address the following

Configuration guide shall be updated with the step by step configuration.

9.14.6 14.6. Future Considerations