Software Design Specification

For

FreeEDR

Submitted by

FreeEDR

|  |  |
| --- | --- |
| **Instructor:** | Professor Greg Hislop |
| **Team Members:** | Bryan Bolesta, Ryan Fiers, Declan Kelly, Matthew Horger, Layla  Phills, Zachary Santoro, Marisa Tranchitella |
| **Cycle:** | 1 |
| **Date Submitted:** | 11/17/2019 |

Document template copyright 2005-2019, Gregory W. Hislop. Version 2.3. Use permitted under Creative Commons license CC-BY-NC-SA. See http://creativecommons.org/licenses/by-nc-sa/3.0/.

Grading Rubric - Design Specification

This rubric outlines the grading criteria for this document. Note that the criteria represent a plan for grading. Change is possible, especially given the dynamic nature of this course. Any change will be applied consistently for the entire class.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Achievement** | **Minimal** | **Exemplary** | **Pts** | **Score** |
| **Content** | Section(s) missing, not useful, inconsistent, or wrong. | Provides all relevant information correctly and with appropriate detail |  |  |
| Introduction |  |  | 10 |  |
| Architectural Description |  |  | 10 |  |
| Interface Description |  |  | 15 |  |
| Detailed Design |  |  | 50 |  |
| **Grammar and Spelling** | Many serious mistakes in grammar or spelling | Grammar, punctuation, and spelling all correct | 5 |  |
| **Expression** | Hard to follow or poor word choices | Clear and concise. A pleasure to read | 5 |  |
| **Tone** | Tone not appropriate for technical writing | Tone is consistently professional |  |  |
| **Organization** | Information difficult to locate | All information is easy to find and important points stand out | 5 |  |
| **Layout** | Layout is inconsistent, visually distracting, or hinders use | Layout is attractive, consistent, and helps guide the reader |  |  |
| **Late Submission** |  |  |  |  |
| **Total** |  |  | 100 |  |

Table of Contents

[1](#_heading=h.2s8eyo1) Introduction 4

[1.1](#_heading=h.17dp8vu) Scope 4

[1.2](#_heading=h.3rdcrjn) Definitions, Acronyms, and Abbreviations 4

[1.3](#_heading=h.26in1rg) Requirements Traceability Matrix 5-6

[2](#_heading=h.lnxbz9) Architectural Description 6

[2.1](#_heading=h.35nkun2) Client Internal with Server Dependencies 7

[2.2](#_heading=h.1ksv4uv) Server Internals separate from Clients 8

[3](#_heading=h.44sinio) Interface Description 9

[3.1](#_heading=h.2jxsxqh) User Interface 9-10

[3.2](#_heading=h.z337ya) Data Interface 10

[3.3](#_heading=h.3j2qqm3) Programming Interface 11

[4](#_heading=h.1y810tw) Detailed Design 12

[4.1](#_heading=h.4i7ojhp) DE1.1 – Dashboard 12-14

[4.2 DE1.2 – Permissions Matrix View](#_heading=h.4i7ojhp) 14-15

[4.3 DE1.3 – Rule Storage Server](#_heading=h.4i7ojhp) 15

[4.4 DE1.4 – Permissions Matrix File](#_heading=h.4i7ojhp) 15-16

[4.5 DE1.5 – Correlation Rule Repository 16](#_heading=h.4i7ojhp)

[4.6 DE1.6 – Event Log 16](#_heading=h.4i7ojhp)

[4.7 DE1.7 – Network Log 16](#_heading=h.4i7ojhp)

[4.8 DE1.8 – Custom Windows Event Log 16](#_heading=h.4i7ojhp)

[4.9 DE1.9 – Internal API’s 16](#_heading=h.4i7ojhp)-18

[4.10 DE1.10 – Endpoint Rule Request & Process Script](#_heading=h.4i7ojhp) 18

Introduction

## Scope

FreeEDR is an open-source endpoint detection and response system for small infrastructure teams and organizations to utilize free of charge. FreeEDR will be completed by June 2020 for a total time allocation of 9 months. Continuous improvements and support of this project will be assumed by Security Risk Advisors, located at 1760 Market Street in Philadelphia, PA after the completion of the initial deployment/release phase.

## Definitions, Acronyms, and Abbreviations

**AD:** active directory, a package of special services to manage permissions and resources on Windows workstations

**API:** application programming interface, technology used for transmitting data between sources such as clients, servers, databases, etc.

**EDR:** also known as endpoint detection and response, a technology used to address the needs for continuous coverage against advanced threats.

**GPO:** group policy object, used when policy settings need to apply to multiple Windows workstations

**Sigma:** generic, open signature format that allows relevant log events to be reported straightforward

**SigmaC:** tool used to translate Sigma format rules to the language of choice

**SIEM:** Security Information and Event Management

**SIRT:** Security Incident Response Team

**SOC1:** also known as a system and organization controls report, used for making sure an organization’s internal control procedures are being properly followed.

**Threat Intelligence Sources:** security feeds from vendors, government / public, and private sources that provide information about known IT vulnerabilities and risks for organizations.

**QA**: Quality Assurance

## Requirements Traceability Matrix

This section maps the relationship between requirement statements and detailed design entities. As such it shows how requirements are covered by the design, and demonstrates the purpose for which design entity exists.

The values in the cells of the table show which requirements provide the purpose for each entity. The cell values are:

* **Blank** – the design entity does not implement any of that requirement
* **P** for Primary - the design entity implements all or most of the requirement
* **S** for Secondary – the design entity implements a smaller but essential part of the requirement

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **DE1.1** | **DE1.2** | **DE1.3** | **DE1.4** | **DE1.5** | **DE1.6** | **DE1.7** | **DE 1.8** | **DE1.9** | **DE 1.10** |
| **R1.1** |  |  | S |  | P |  |  |  |  |  |
| **R1.2** |  |  |  | P |  |  |  |  |  |  |
| **R1.3** |  |  | S |  |  |  |  |  |  |  |
| **R1.4** |  |  |  | P |  |  |  |  |  |  |
| **R1.5** |  |  |  | P |  |  |  |  |  |  |
| **R2.1** |  |  | S |  |  |  |  |  | P | P |
| **R2.2** |  | S |  |  |  |  |  |  |  |  |
| **R2.3** |  |  |  |  |  | S |  |  | P | P |
| **R2.4** |  |  |  |  |  |  |  |  | P |  |
| **R2.5** |  |  |  |  |  |  |  |  | P |  |
| **R2.6** |  |  |  |  |  | S | S | P |  |  |
| **R2.7** |  |  |  |  |  |  |  |  | P |  |
| **R2.8** |  |  |  |  |  |  |  |  | P |  |
| **R2.9** |  |  |  |  |  |  |  | S |  | P |
| **R3.1** |  |  | S |  |  |  |  |  |  |  |
| **R3.2** |  | P |  |  |  |  |  |  |  |  |
| **R3.3** | P |  |  |  |  |  |  |  |  |  |
| **R3.4** | S | P |  | S |  |  |  |  |  |  |
| **R3.5** | P |  |  |  |  |  |  |  |  |  |
| **R3.6** | P |  |  |  |  |  |  |  |  |  |
| **R3.7** | P |  |  |  |  |  |  |  |  |  |
| **R3.8** | P |  |  |  |  |  |  |  |  |  |
| **R3.9** | P |  |  |  |  |  |  |  |  |  |
| **R4.1** |  |  |  |  |  | P | P |  |  |  |
| **R4.2** |  |  |  |  |  |  |  | S | P |  |
| **R4.3** |  |  |  |  |  |  |  | S | P |  |
| **R4.4** |  |  | S |  |  | P | P |  |  |  |
| **R4.5** |  |  |  |  |  |  |  |  | P |  |
| **R4.6** | S |  |  |  |  |  |  |  |  |  |
| **R4.7** | P |  |  |  |  |  |  |  |  |  |

Figure 1 – Traceability between Functional Requirements and Design Entities

# Architectural Description

FreeEDR operates on a client-server foundation and has version dependencies on each aspect of the architecture. The client workstations within FreeEDR are running Windows operating systems, starting with version 7 and can be as recent as Windows 10. These workstations have no security restrictions to run PowerShell scripts as they are solely used to query the Windows Event Log. Our server is operating on the latest release of Windows Server 2016, version 1607. The version of IIS installed on this server will be 10.0 due to its unique advanced settings and functionality with ASP.NET API’s. For communicating results from the server, an instance of Microsoft Exchange O365 is utilized as the primary tool since Microsoft recommends that all new projects and services utilize their Exchange Web Services (EWS) platform to move away from legacy 2013 and 2016 environments.

For the purposes of abstracting off information, the two diagrams presented below represent interactions between the clients internally and their dependencies on the server, as well as the actions performed by the server separately from the client.

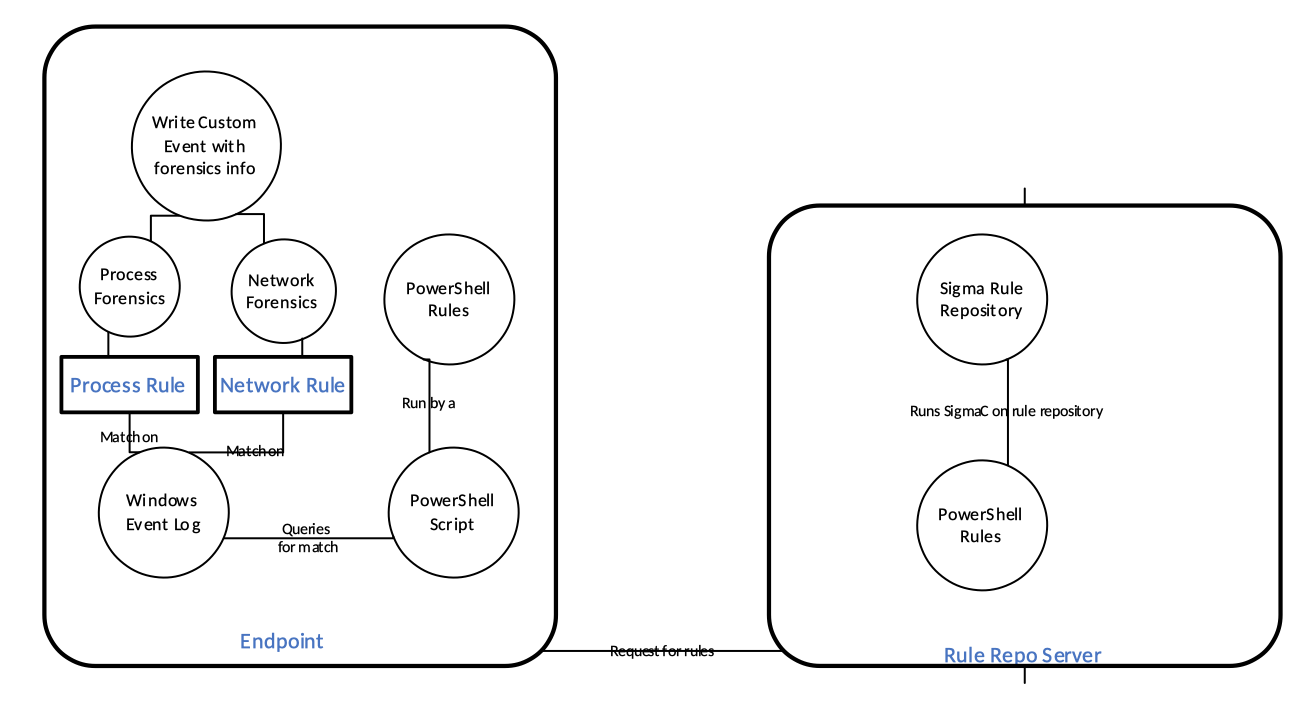


Figure 2 - Rule Repository Server Communication with Endpoints

In this diagram, the client which is represented as an endpoint is detailed on the left side while the server which stores the rules repository is located on the right. On the server side, PowerShell rules execute SigmaC on the Sigma Rule Repository object. The client then will request for any rules from that server upon the client doing the analysis. On the client side, these PowerShell rules are run to query for a match based on what event is produced on the client’s workstation. FreeEDR gathers custom event information, categorizes the event either as a process or a network event, and then performs forensics on the event. Depending on the result of forensics, our process and network rules then write the information to be stored out to a custom Windows Event. By abstracting process and network categories to perform certain rules, this procedure allows for the FreeEDR to provide more detailed forensics information for the custom event.

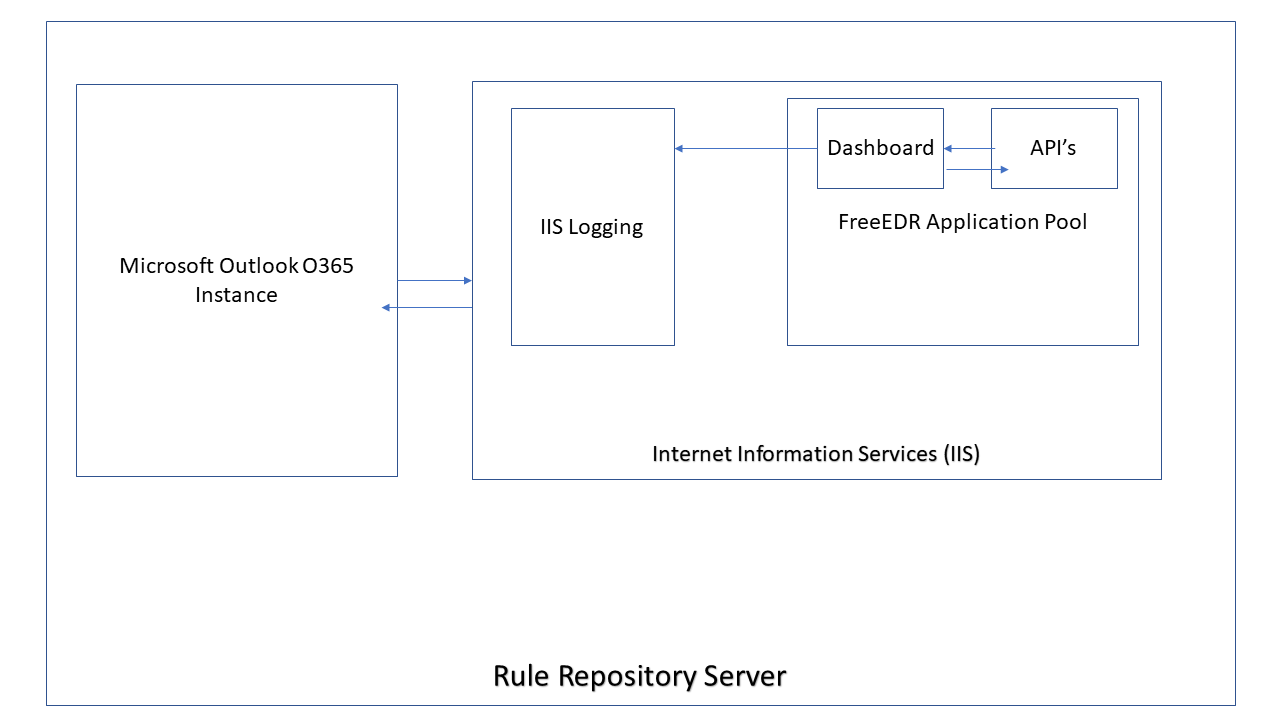


Figure 3 - IIS Configuration for Dashboard / other API’s

In order for our internal application endpoints to communicate within FreeEDR, IIS must be setup on the same server as the rule repository storage object is located. In IIS, we must have an application pool dedicated to serving up applications assigned to a specific Active Directory user. By having both the Dashboard and API assigned to a specific AD user, FreeEDR is able to easily monitor all endpoints and traffic to the instances via %systemroot%\system32\logfiles for that specific user. Our application pool connects to the built-in IIS logging system that Dashboard Infrastructure managers are able to view in case there are any issues with the product. Finally, the IIS instance running on the Windows server is able to connect to a Microsoft Outlook O365 instance, provided that the Dashboard Infrastructure Manager configures the Exchange Web Service (EWS) autodiscover url in the app.config section of every API and Dashboard deployment. A sample autodiscover url is one like so:

[***https://mail.domain.com/ews/exchange.asmx***](https://mail.domain.com/ews/exchange.asmx)

By having the autodiscover URL in the app.config, the Dashboard is independent of changes to the EWS instance and only relies on a single entrypoint, rather than having multiple dependencies within the code itself and relying on emergency redeployments every time the O365 instance changes.

# Interface Description

## User Interface

The system’s user interface will display information in a dashboard format coded in React with component implementations from Google’s Material UI and MUI Datatables. The dashboard is sectioned off into 5 distinct sections.

**3.1.1**  Home:

The Home page is intended to display a custom data view where clients can see what reporting information is important to each respective user. The home page will rely on an internally developed engine to filter specific reports and information to allow for users to pick and choose data grid views. Some of the visual aides that will be implemented are tabular, bar charts, and typical XLSX graphs.

**3.1.2** Reporting:

The Reporting page is intended for Incident Response team members to ensure that security alerts are being properly reviewed, responded, and analyzed. This page is critical for Incident Response team members as they are able to view all the reports they can generate to see what events are happening more or less frequently and adapt their practices to adjust for these client actions. The reporting page has many report generation options, which each will offer a destination for the report to be sent or saved to. Some of the few reporting options are:

* Security Event Log - This report generates a list of all security events throughout all of FreeEDR. This report can be generated daily, weekly, or for any historical period.
* User Event Log - This report generates a list of all security events produced by a client endpoint. This report, like most, can also be generated for a given span of time.
* Process / Network Event Log - This report generates a list of all process or network security events depending on a user’s selection.
* Trend Log - This report generates a historical model of when security events are triggered

**3.1.3**  Audit:

The Audit page is intended for system auditors to have view access of all system and user logs, as well as perform historical and up-to-date tasks when performing auditing procedures. The primary function of the audit page is to allow for system auditors to generate audit reports within a given period of selected time in order to ensure that the system is compliant and all best practices are being followed. The audit page also produces views that system auditors to see when previous SOC1 reports were generated and view historical trends of when audits were performed. This audit page is customizable so that certain auditors in a team may have ease of access to generate reports that they normally are assigned to.

**3.1.4**  Support:

The Support section is intended for users of the dashboard to seek help when the dashboard is not functioning properly, or data seems invalid. There are three key components to the support section:

* Contact Incident Support - This page is intended to provide information a user would need to get in touch with an Incident Support member. This will be a listing for their office number. Users should use this page when they suspect they have encountered a critical security issue.
* Support Hotline - This page is intended to allow users to contact a member of the Dashboard Infrastructure team if they suspect something is wrong with the system. This includes, but is not limited to, invalid data, permissions, and operation functionality.
* Submit a Ticket - This page is intended to allow users to request new features for the dashboard, or ask general inquiry questions for the support team.

**3.1.5** Admin:

The Admin section is intended solely for support staff of the dashboard, which includes Incident Support and Dashboard Infrastructure team members. There are two key components of the admin section:

* Permissions Matrix - This page is intended for Dashboard Infrastructure support members to be able to view, add, and remove access permissions for users in regards to the dashboard itself. The matrix itself will be built inside a MUI Datatable to allow for support members to grant access remotely via a mobile browser.
* View Rule Repository - This page is intended for Incident Support Managers to quickly view the rule repository and the secured contents. This page does not have any functionality to modify the rule repository or contents, as those changes must be made external of a front-facing application.

## Data Interface

FreeEDR accepts certain files from external third-party threat intelligence sources. The file is classified as a Threat Intelligence Rule, and has a strict classification of a data type. These files are YAML files which contain rule logic in an expected Sigma format for our system to process. These Threat Intelligence Rules are then stored in FreeEDR’s Rule Repository which are then translated to PowerShell queries to be executed.

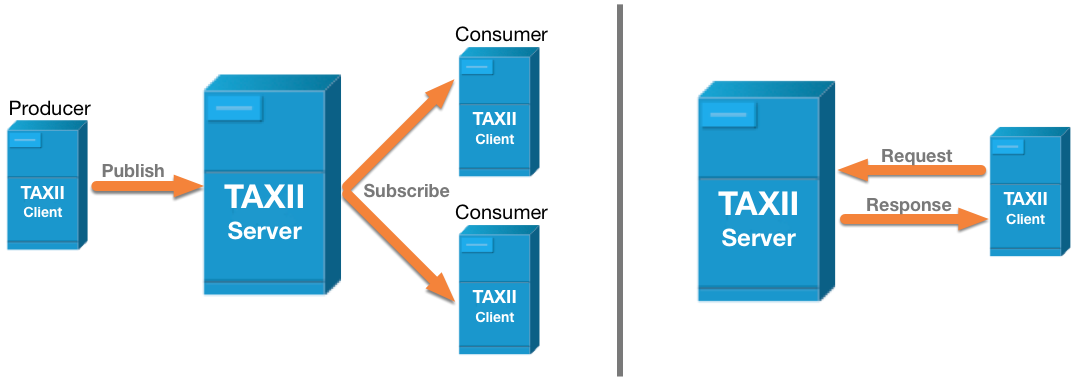


Figure 4 - A simple diagram showing the interactions between TAXII and Clients. FreeEDR is a consumer of information from a TAXII server, and makes requests to fetch information.

## Application Interfaces

FreeEDR utilizes TAXII (Trusted Automated eXchange of Indicator Information), which is a free and open transport mechanism that standardizes the automated exchange of cyber security and risk information. FreeEDR makes requests to a TAXII server in order to pull down STIX (Structured Threat Information Expression) data. This application interface allows for TAXII to determine where to store this Threat Intelligence Rules so that FreeEDR can process the data interface above.

# 

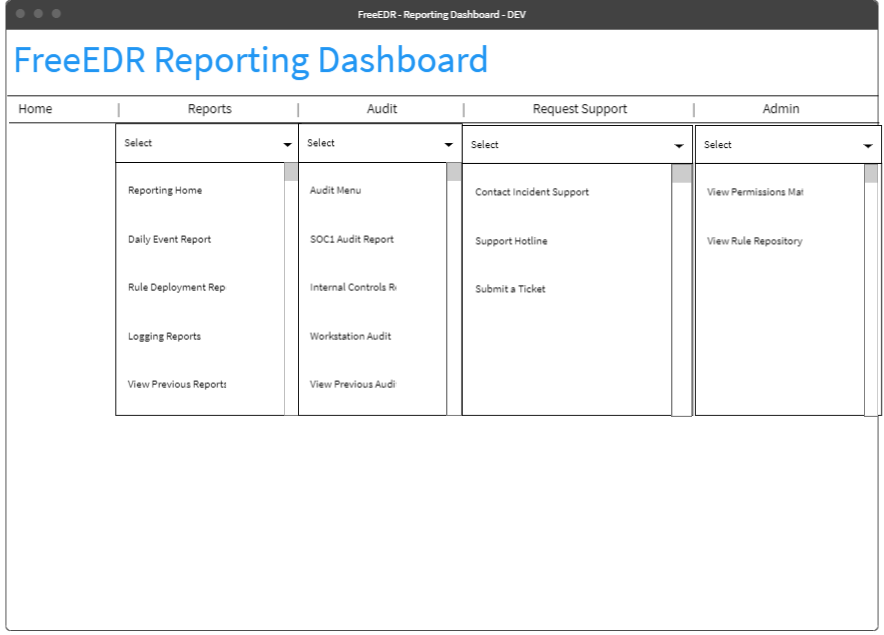
# Detailed Design

## DE1.1 - Dashboard

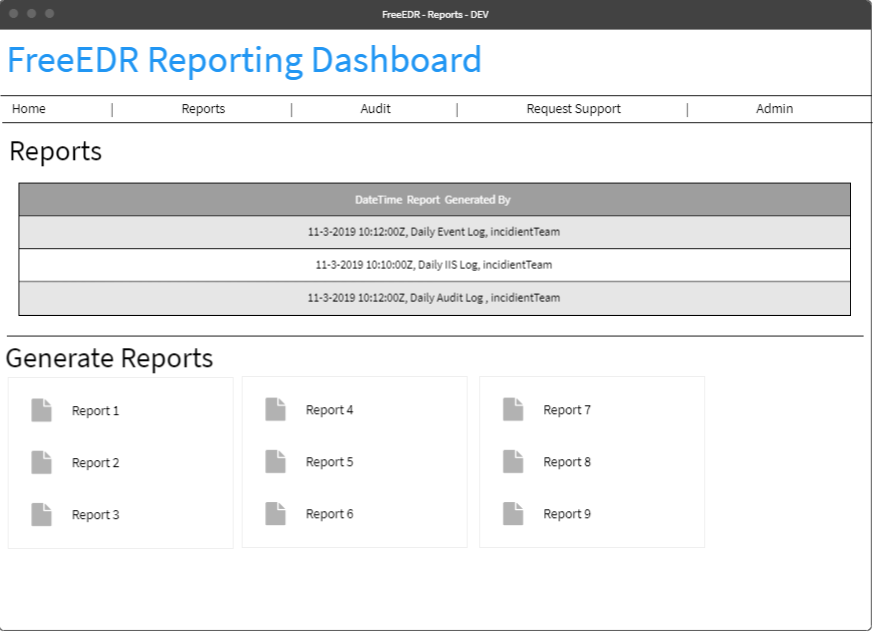
**Type**: Screen

**Description**: The dashboard is the home screen of the system’s user interface. Here the user is able to view previously generated reports and produce fresh reports.

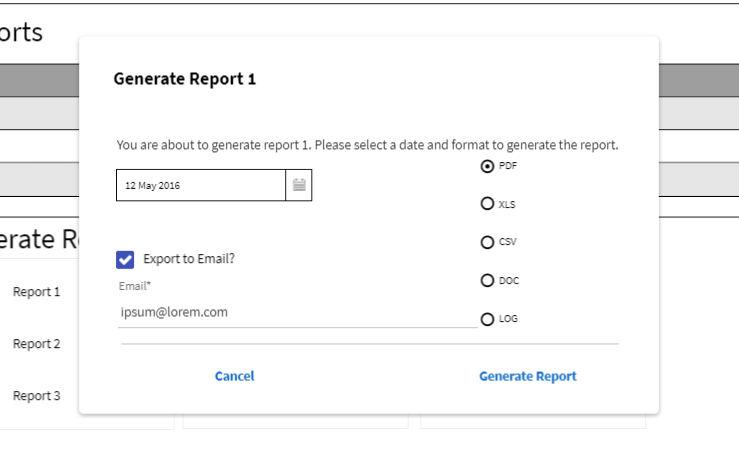
**Mockups:**



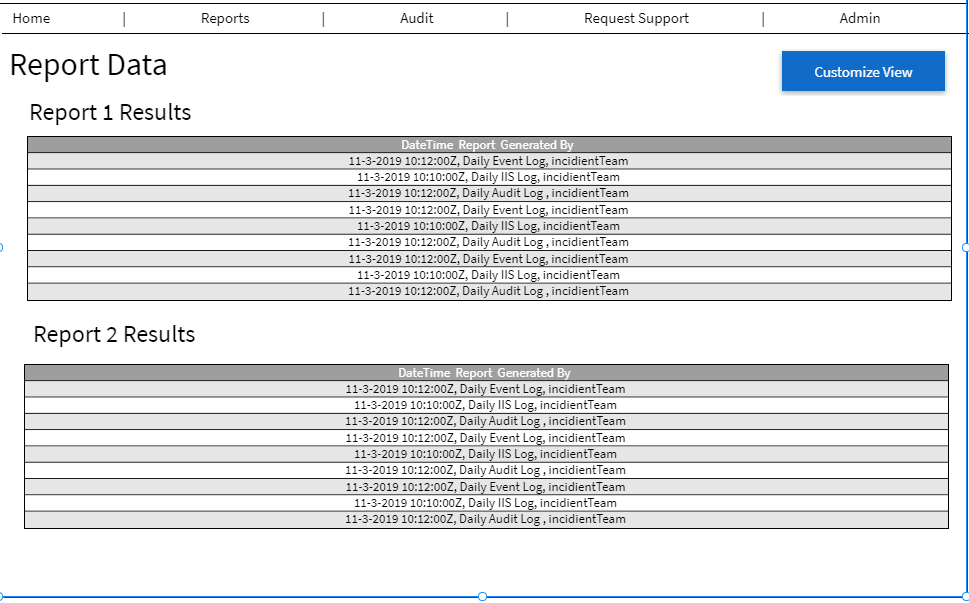
This mockup demonstrates the drop-down functionality for the menu dashboard. All clients will have read access to this page, but will be unable to target certain options (eg Admin).



This mockup represents the ability for users to see recently generated reports, as well as the option to trigger the generation for new reports.



This mockup represents the dialog box presented to users when they want to generate a report. Date range is only available for specific reports and not listed here.

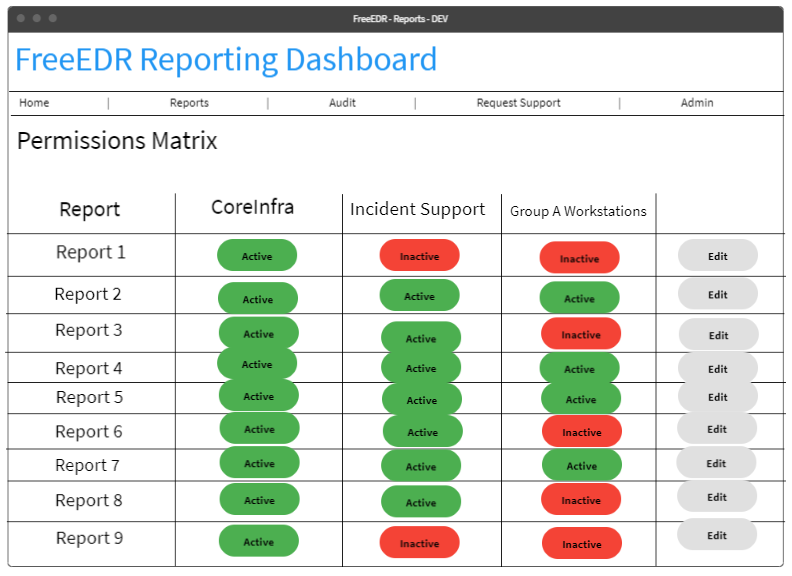


This mockup shows one possibility for a user to customize their homepage data view. Other options include bar graphs and other React-MaterialUI possibilities. Tabular is shown here as a limitation of the mockup design software.

## DE1.2 - Permissions Matrix View

**Type**: Screen

**Description**: The permissions matrix view will be accessible from the dashboard screen and will allow the dashboard infrastructure manager to view and adjust client permissions.

**Mockup:** 

This mockup shows the permission matrix that can be customized for user permissions to the dashboard and reports. This component will be utilizing MUI Datatables for appropriately lined up chips and data grid operations, as the wireframe tool shown here does not allow for snap-to-grid functionality.

## DE1.3 - Rule Storage Server

**Type**: Server

**Description**: The server that stores all rule data objects. This server also hosts the Dashboard and API’s in IIS.

**File Format:** NTFS

## DE1.4 - Permissions Matrix File

**Type**: Data file

**Description**: The file that stores the permissions that the dashboard infrastructure manager creates.

**Format:** JSON

**Example:**

[ {“report1: {“CoreInfra”: active, “IncidentSupport”: inactive, “GroupA”: inactive}}, {“report2: {“CoreInfra”: active, “IncidentSupport”: active, “GroupA”: active}}]

## DE1.5 - Correlation Rule Repository

**Type**: Data repository

**Description**: The correlation rule repository houses rules. Rules can contain the following attributes: title, status, description, author, references, logsource, detection, false positives and level. These attributes are customizable.

## DE1.6 - Event Log

**Type**: Data file

**Description**: The event log records all captured events generated by IIS functions. The attributes for an event include: event, event type, event ID, source, log location, data and time, task category, keywords, computer, user, opCode and more information.

## DE1.7 - Network Log

**Type**: Data file

**Description**: The network log stores data that is created when traffic occurs between clients and IIS operations. Network records contain the following attributes: name of machine, user account, date time of request, IP address, HTTP response status code, headers, requested IP addressed and number of packets transmitted.

## DE1.8 - Custom Windows Event Log

**Type**: Data file

**Description:** The custom Windows Event is a Windows Event that has been enriched with forensics data for the process or network connection. The forensics information specifies how trustworthy the process or network connection.

## 

## DE1.9 - Internal API’s

**Type**: C# WCF RESTFul API

**Description**: In order to satisfy requirement R3.1, FreeEDR will provide an API written in C# and protocoled in Windows Communication Foundation (WCF) in order to transmit data between the dashboard and rule repository.

* + This API will be exposed via RESTFul calls instead of the default SOAP configuration set by WCF.
  + This API must be deployed on an Internet Information Services (IIS) instance in order to assign a service account for security access. This service account is what the Dashboard Infrastructure Manager will use to permission users to view specific reports and actions via requirement R3.5.
  + Endpoints:
    - GetReportOptions()
      * This endpoint will return the list of report options that can be generated.
    - GetHistoricalReports(DateTime dt)
      * This endpoint takes in a date time object and returns the reports generated for the given date.
    - GetReport(Report r)
      * This endpoint generates a given report object for the current date in the default format of PDF.
    - GetReport(Report r, DateTime dt)
      * This endpoint is an overload of the default endpoint to generate a report object. This endpoint allows for reports to be regenerated for previous dates.
    - GetReport(Report r, FormatObject f)
      * This endpoint allows for users to specify which format they want their requested report object to be returned as with the default date.
    - GetReport(Report r, DateTime dt, FormatObject f)
      * This endpoint allows for users to specify which format and date they want their requested report object to be returned as.
    - GetReportRange(Report r, DateTime start, DateTime end)
      * This endpoint generates a list of reports of a given report object within a start and end date.
    - ExportReport(Report r, string recipient)
      * This endpoint forwards a generated report to a requested email address.
* In order to satisfy requirement R3.7, FreeEDR must supply a MailService API that restricts sending reports externally to recipients.
  + This API corresponds with the report generation API in that it is a connected service referenced.
  + This API will be connected using Microsoft Outlook O365 to provide the capacities to send emails.
  + This API must be configured with an organization’s auto discover URL in order to properly connect to the running instance.
  + Endpoints:
    - SendMail(string sender, string recipient, string subject, string message)
      * This endpoint will send a specific HTML mail message from the requested sender to the recipient.
      * This recipient parameter could take in multiple contacts separated by a comma.
    - SendMail(string sender, string recipient, string subject, string message, string attachment)
      * This endpoint is an overloaded endpoint that allows for an option to send an attachment.
      * The attachment must be a valid folder path.
      * The attachment must be accessible by the service account assigned to the application pool in IIS.
      * The attachment must not be larger than Outlook’s size requirements (>10mb).
* In order to satisfy requirement R3.8 and R3.9, FreeEDR will implement a ‘heartbeat’ API for the dashboard to utilize.
  + This API will be written on a low-level scale in order to promote fast response times to fetch new data requests.
  + This API needs to be separate from the other main API in order to not spawn multiple concurrent resource allocations which will drawback from performance.
  + Endpoints:
    - CheckAlive()
      * This endpoint checks to make sure the heartbeat service is alive.
    - FetchData()
      * This endpoint checks to make sure that there is no fresh data that needs to be populated before a report is generated.
    - Restart()
      * This endpoint forces all concurrent connections to drop communication.

## DE1.10 - Endpoint Rule Request & Process Script

**Type**: Powershell script

**Description**: The endpoint rule request script will be assigned to each endpoint. Its purpose is to reach out to the rule repository and request any new rules and store them in C:\FreeEDR. Then, it runs the rules to see if there is a matching windows event. If the rule matches on a process event, then the script will gather forensics information about the specific process. If the rule matches on a network event, the script will gather forensics information about the network connection.