SOFTWARE REQUIREMENTS SPECIFICATION

for

FreeEDR

Version 2.0.0 approved

Prepared by:
Bryan Bolesta
Ryan Fiers
Declan Kelly
Matthew Horger
Layla Phills
Zachary Santoro
Marisa Tranchitella

Team: FreeEDR
October 27, 2019

Contents

Revision History

Name	Date	Reason For Changes	Version
1.0.0	18–10–19	Initial Structure	mhorger
1.1.0	21–10–19	Design Constrains	dkelly, bbolesta
1.1.1	24–10–19	User Characteristics	lphills
2.0.0	27–10–19	Draft Submission	mhorger

1 Introduction

1.1 Purpose of Document

The purpose of this software requirement specification document is to outline the functionality and features of our proposed project. We fully expect that this document will be detailed enough about how it will be expected to perform, maintained, and potentially developed further. This document will not describe how the system works in detail or how infrastructure teams should utilize the project. The intended audience of this document is three-fold: first is developers who can review the project's requirements to see where time and research efforts can be utilized to improve the project. Second is UAT testers who can test our product when deployed in an environment to find potential security risks or unexpected bugs that stem from requirements. Lastly is infrastructure team members of an organization (end-users) interested in implementing our system to enhance their security footprint of workstations.

1.2 Project Scope

This project will consist of creating a open-source endpoint detection and response system for small infrastructure teams and organizations to utilize free of charge. The project 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 after the completion of the initial deployment/release phase.

1.3 Definitions, Acronyms, and Abbreviations

1. AD: active directory, a package of special services to manage permissions and resources on Windows workstations
2. API: application programming interface, technology used for transmitting data between sources such as clients, servers, databases, etc.
3. EDR: also known as endpoint detection and response, a technology used to address the needs for continuous coverage against advanced threats.
4. GPO: group policy object, used when policy settings need to apply to multiple Windows workstations
5. Sigma: generic, open signature format that allows relevant log events to be reported straightforward

6. SigmaC: tool used to translate Sigma format rules to the language of choice
7. SIEM: Security Information and Event Management
8. SIRT: Security Incident Response Team
9. QA: Quality Assurance

1.4 Overview

Organizations that don't have a significant security budget can find it difficult to include workstations in their monitoring scope. Forwarding logs from all the organization's workstations can be expensive because most SIEMs are priced based on log ingestion and tools such as EDR are just as expensive. This project aims to setup a series of scripts which will allow organizations who don't have the ability to purchase or implement an enterprise solution. This solution is aimed at devices running Windows as it relies on querying Windows events using PowerShell.

2 Overall Description

2.1 Product Functions

1. Provide a series of PowerShell scripts to generate security alerts by querying Windows events on Windows endpoints.
2. Create a repository server to store detection rules and be used in the process of updating and delivering new rules to Windows endpoints.
3. Generate various process and network detection rules to used by our scripts to gather event forensics and generate Custom Windows Events.
4. Provide methods for forwarding newly generated Custom Windows Events to SIEM solutions, and for setting up email and message alerts when these Custom Windows Events are triggered.
5. Provide an interactive dashboard in order to produce reports, logs, and other auditable information detailing specific time-stamped information.

2.2 User Characteristics

2.2.1 Script Manager

This user is responsible for the retainment, management, and deployment of the provided PowerShell scripts. The script manager must have the privileges and means to deploy and run the scripts on Windows endpoints across the organization. The main function of this user is to deploy the series of scripts to any in-scope endpoints, retain the scripts, and re-deploy scripts when necessary.

2.2.2 Incident Response Manager

This user is responsible for ensuring that alerting methods are set-up and properly configured. This user is also responsible for reviewing, responding, and analyzing security alerts. The Incident Response Manager must have view access to the platform that the alerts are being sent (SIEM, Email, or Message). The main function of the Incident Response Manager is to review and respond to the security alerts generated by the FreeEDR scripts.

2.2.3 Incident Response Supporter

This user is responsible for aiding the Incident Response Manager in responding to incidents in real time. This user must have the privileges and means to view the reports curated specifically for less technical people. The main function of the incident response supporter is to follow basic instructions to complete tasks that will offset the responsibilities of the Incident Response Manager during an incident response.

2.2.4 System Auditor

This user is responsible for auditing the functionality and use of the system. The auditor must have view access of all system and user logs. This user must also have access to any previous audit reports. The main function of this user is to ensure that best practices were followed, the system is being used as effectively as possible and to report any compliance issues discovered to the appropriate parties.

2.2.5 Dashboard Infrastructure Manager

This user is responsible for maintaining and deploying future releases of the Audit Dashboard for auditors to view. The infrastructure manager must make sure that the dashboard is consistent with the information that is being produced for the system auditor, and have continuous support capacities if some errors were to occur with the reporting tool.

3 Specific Requirements

3.1 Functional Requirements

R1. Server - Rule Storage

- R1.1. Server will store correlation logic for process and network events
- R1.2. Server will allow read access for clients in the deployment
- R1.3. Server will connect with threat intelligence sources to discover new correlation rules
- R1.4. Server will allow the system administrator to update correlation logic

R1. Client - Rule Processing

- R1.1. Client will communicate with the Rule Storage Server to pull down correlation rules.
- R1.2. Client will check the rules against the event log every hour.
- R1.3. Client will connect with application APIs to perform network and process forensics on an event.
- R1.4. Client will store process and network forensic information in a single location.
- R1.5. Client will store forensic information for an amount of time set by the system administrator.
- R1.6. Client will forward forensic event information to a central location to be actioned by an analyst.
- R1.7. Client will perform a different type of forensics for process and network events.
- R1.8. Client will be able to identify the difference between process events and network events.

R1. Dashboard

- R1.1. Dashboard must be hosted on the same server used for rule storage due to security. This is a design constraint due to the way the dashboard will process information.
- R1.2. Dashboard must be accessible to all clients to view previously generated reports as well as produce reports.
- R1.3. Dashboard must be flexible for modifications in order to add/remove reports as needed by auditors.

R1.4. Dashboard must allow clients to select a range of dates for report generation.

R1.5. Dashboard must allow clients to select a format to download their generated reports.

R1.6. Dashboard must have a responsive algorithm that allows for regeneration of reports once fresh data is loaded in a folder (see Data Requirements further on the data types).

R1.7. Dashboard should be accessible 90% of normal business hours, with the exception being disaster recovery downtime / failover procedures.

R1.8. Dashboard must have an option to export a report to send via interdepartmental communication (email, IM, etc).

3.2 Non-Functional Requirements

N1. Hardware

N1.1. The system will only use hardware in compliance with SRA's minimum security requirements.

N1.2. The system will be stored on a secure and segmented part of the CyberDragon's server until system ownership is transferred to SRA.

N1. Network

N1.1. The system will not use or interact with the SRA network in any way that violates any of SRA's privacy policies.

N1.2. The system will have 98.9% availability on the network during standard business hours.

N1.3. The system will return data within 15 seconds of the customer's request.

N1. Deployment

N1.1. The system will be able to handle all exceptions created by erroneous user input.

N1.2. All deployed versions of this system must have passed all QA / UAT tests.

N1.3. The system will consume at most 250 megabytes of memory.

N1.4. The system's Mean Time to Change (MTTC) for issues will be ¡ 3 person days.

N1.5. The system will satisfy all of Nick Ascoli's, our external stakeholder for this project, UI preferences when system ownership is transferred to SRA.

3.3 Data Requirements

D1. Clients will not be able to write to the rules repository.

D2. Clients will be presented with data on endpoint events (i.e. registry modifications, cross-process events, file executions, network connections).

D3. Client will have access to relevant data from threat intelligence platforms.

D4. Dashboard should have access to all necessary data in order to produce reports. This data includes forensic event information, user machine configuration, and standard log outputs.

D5. Dashboard must have a service account associated with operations in order to track requests, actions, etc in regards to manipulating data.

D6. Data for dashboard must be sourced via a folder for easy data storage and retrieval.

D7. Data used for report generations must not have a lock placed on the file in order to allow for concurrent operations.

D8. Data displayed in dashboard must be in a concise, readable format with an option for details to be viewed separately.

D9. Data transmitted via every API must be under the proper protocols for security (POST) and sensitive information must be encrypted before transit.

D10. Data must be kept for up to 5 years. Data past 5 years is outside the jurisdiction of auditable actions.

3.4 Design Constraints

DR1. Correlation rules should be written in Sigma to allow sharing through threat intelligence platforms such as Threat Alert Logic Repository (TALR)

DR2. Rules must be translated to PowerShell Get-Event Queries as this is the supported language in Sigma.

DR3. System should be deployed in a Windows environment due to the reliance on Get-Event PowerShell queries.

DR4. Clients must deploy Syson to capture the appropriate events for the system to perform properly

Bibliography