Project Charter

Cyberthreat Insight Discovery & Visualization Tool

Webroot

Prepared by CU Boulder Capstone Team

October 10, 2019

Revision History

Charter Revision Register: This section is used to document any changes and serves to control the development and distribution of revisions to the project charter. It should be used together with a change management processes as appropriate, and housed in the formal project repository. Note: Best practice is to save the original baseline version as a separate file so that an accurate history of the original document remains available for comparison.

Change Description	Approved by	Date of Revision
Initial Approval	Eric Klonowski	10/10/19

Approval: ___________________________

Accountable Executive/Project Sponsor

i. Revision History **1**

ii. Table of Contents **2**

1. Introduction **3**

Executive Summary 3
Business Problems/Opportunities 3

2. Objectives and Scope **4**

Business Objectives 4
High Level Requirements 4
Constraints 5

3. Project Scope **6**

In Scope 6
Out of Scope 7

4. Project Approach **8**

General Approach – Solution Delivery Process 8
Assumptions 8
Project Risks and Issues 8
Project Changes 8

5. Project Plan 9

Development Stack 9
Timeline 10

6. CU Group Roles & Responsibilities 12

7. Key Stakeholder Roles & Responsibilities 13

1. Introduction

Executive Summary

- The CU Boulder team will produce an interface that allows for Webroot's researchers and engineers to construct graphical visualizations of threat content based on Indicators of Compromise (IoC).
- The tool's user interface will conform to industry tools such as Virus Total while the data represented will come from data generated by Webroot's threat assessment modules.
- Researchers will be able to query specific artifacts, and optionally extend their search by adding more queries to their graph interface, while being presented with all connections in the process. The graph will be saveable and loadable, with the ability to view IoC details for each node.

2. Objectives

Business Objectives

- Create an internal tool that Webroot engineers and researchers can use to track threats and inform decisions about future defense modules.

High Level Requirements

- Nodes in the graph will be represented abstractly due to the variable nature of IoCs which are inherent to a document, or a document's history. Connections between IoCs can also be generated from executing binaries. As of the writing of this document, the visualization tool will be capable of representing IoCs including the following list:
    - File hash checksums
    - Fuzzy hash similarity
    - Registry entries
    - Created/Written files & locations
    - Hosts contacted
        - IP / Port
        - DNS
    - File string content
    - similarities between behaviors of binary files

- The graphing interface will have the following main features
    - Ability to query Webroot's DynamoDB
    - Ability for users to physically adjust position / orientation of a graph nodes
    - Ability for nodes to hold hidden connections that can be shown by user choice
        - Implies an ability for nodes to hold references to all possible connections
    - Ability for the graph to change dynamically as new nodes and connections are added by the user
    - Ability for users to create new connections between nodes
    - Ability for users to annotate graphs with (non graph connection) notes
    - Ability to save and load a graph with all user changes made
    - Ability for a user to open a saved graph from another user

- Data in the form of IoCs will be stored in a DynamoDB database managed by Webroot. New data entries into this database trigger a sequence of events that result in autogenerated final tables filled out by Webroot's modules. Our tool will query only these final tables.
- The structure of these tables is subject to change. As such, our tool will have a modular implementation that allows the code to be easily maintained to account for changes in database structure without altering the front end graphical application.

Constraints

- Webroot’s current database service through Amazon Web Services (AWS) charges per query. Our design will minimize the number of queries made to Webroot’s database by storing data for graph visualization and loading in a separate database. At this time, the separate database will also be an AWS database.
- To illuminate the scalability of the tool, our team will provide a cost analysis based on the number of queries generated and pricing models for various database solutions available from Amazon.

3. Scope

Project Scope

The goal of this project will be to create a web based user interface that displays a graph consisting of many nodes with visual links to show relationships between nodes. The interface will mimic much of the functionality and workflow of Webroot's current too, VirusTotal.

In Scope

- The graph will be the central part of the interface and must have the following attributes:
    - The graph must be expandable and dynamic. That is, there should be some way to add additional nodes and relationships to an existing graph.
    - The orientation of nodes in the graph must be customizable. Users should be able to rearrange the positions of nodes in the graph. The positions of the nodes should also save between sessions, so that if a user returns and loads a saved graph the nodes will appear in the positions where they were last.
    - The graph must be savable and shareable between users. While managing user authentication and accounts is **NOT** within the scope of this project, users should be able to save a graph and load that graph when returning in a new session.
    - Users must be able to annotate the graph in some form. Annotations must save along with the graph state.
- Each node in the graph will have the following capabilities:
    - Nodes must be abstract and extendable. Nodes should be implemented with the assumption that the possible types of nodes will be expanded in the future, and any implementation should allow the addition of node types without significant changes to the source code structure.
    - Nodes should be selectable and display additional information about the object represented by the node. This information will be gathered from the database.
- Part of the project will be to create a database that receives data from the Webroot data pipeline and store that information in a way that is accessible by the graph interface. The database implementation must have the following attributes:
    - The database must rely on an AWS database technology. Any database implementation should be deployable on Webroot's AWS stack and be reasonably cost effective. DynamoDB will be used as a baseline to evaluate any cost difference.
    - Users should not be able to modify the contents of the database. All node information should come directly from the Webroot pipeline, and should only be modified through that pipeline.

- In terms of stack architecture, the full stack will be in scope except for hosting the final product (on servers, or integration with Webroot's current stack), and management of Webroot's current DynamoDB which generates data.

Out of Scope

- Anything that extends beyond the agreed upon project scope and technologies should be assumed to be out-of-scope; though this does not entail that no propositions or technologies outside of the scope can be used.
- Assumptions**:** The project may eventually include a suite of features that we do not currently plan to integrate:
    - We will enable multiple users to share saved graphs. A graph here refers to an annotated, or user-expanded version of the graphs generated from basic queries. As such, we will design the tool to allow multiple users, and to save/load graphs based on a user key. However, it is out of the scope of this project to handle user-authentication and management.
    - We will not currently plan to host or deploy the final version of the product. Dockerizing the tool is within the scope of the project, however.

4. Project Approach

General Approach – Solution Delivery Process

- The CU development team will use an Agile workflow approach
- Each sprint will work toward the idealized "Schedule" listed in the Project Plan section below
- The Agile workflow will allow minor adjustments to be made during throughout the project
- Weekly project status reports will be made available to keep everyone apprised of the project status
- Any proposed major alterations to the development cycle should first be reviewed by the development team, then approved by Webroot and vice versa

Assumptions

- Once the project is delivered, Webroot will perform all required upkeep and maintenance
- The development stack is left up to the CU development team
- The database that receives data from the Webroot data pipeline and store that information in a way that is accessible by the graph interface will rely on AWS technology

Project Risks and Issues

- AWS costs can vary depending on final implementation of pipeline and how queries are implemented
- The project needs to allow for possible changes with the DynamoDB database managed by Webroot that is still in development
- Threat Analysis Pipeline database is new technology that is still undergoing changes

Project Changes

- Changes to project scope and requirements will be reviewed by the requirements lead, Warren Payne. Warren will initiate a discussion with the project team regarding the potential impact of the change on schedule and resources, then provide a decision for approval/disapproval. All scope/requirements changes will be documented and signed off on by both parties.

5. Project Plan

Development Stack

- The team plans to develop the interface as a web app similar to current industry products like Virus Total.
- Project architecture plans include using the following libraries/languages:
    - Node.js for communication between the database and web app
    - React for frontend UI design
    - An AWS NoSQL database to store IoC database
        - Which AWS solution we choose will depend on cost analysis
    - A JavaScript graph visualization library to construct the IoC relationships
        - The framework we choose will be included in the project proposal

Schedule

Project Phase:	**Deliverables/Milestone:**	**Baseline Time Box Dates (Subject to Change)**	
		Start	**Finish**
Initiate	● Designate time(s) for weekly team meetings ● Meet with Webroot primary contacts & establish meeting schedule ● Study project background ● Assign team roles ● Establish project budget ● Initiate project	9/16/19	10/11
		Start	**Finish**
Plan	● Complete Project Charter ● Develop rough development cycle o Define technologies o Define container/component architectures ● Submit (to Webroot) a project proposal ● Familiarize with chosen tech stack ● Develop detailed development cycle & finalize technologies ● Develop work (schedule)	10/14	12/13
		Start	**Finish**
Execute	● Finalize detailed requirements ● Complete API support o Authenticated access to Threat Analysis Pipeline database o Query of artifacts o Compound IoC queries o Unit tests ● Complete User Interface Support o Display of queried Artifact metadata o Display of queried IoC data	12/16/20	4/24/20

	o User creation of graph nodes o Vector linking of graph nodes (based on correlating queries) ● Complete API + Database support o Storage of user created node graphs o Retrieval of user created node graphs o Storage of user defined tags o Sharing of node graphs between users o Unit Tests ● Complete testing		
Close	● Final test cases ● Ensure final deliverable meets predetermined requirements ● Receive approval to close project	**Start** 4/27/20	**Finish** 5/11/20

6. CU Group Roles & Responsibilities

Architecture Lead Alex Burnley	● Manages the relationships and interactions between selected frameworks and libraries
Communications Lead **& Team Lead** Kamiar Coffey	● Liaison between Webroot and capstone group ● Primary point of contact for project related logistics ● Schedules meetings and provides general project updates
Requirements Lead Warren Payne	● Manages scope changes during the course of the project ● Ensures the group's deliverables and the sponsor's expectations align
Testing Lead Alex Markley	● Ensures all testable code pushed has appropriate unit and integration tests ● Creates automated testing capabilities with a CI tool ● Ensures tests have adequate coverage of the codebase ● Assigns bug fix stories for failing tests
Deployment Lead Connor Guerin	● Manages containerization of the project ● Coordinates with Architecture Lead to ensure that all components are smoothing integrated in a final product
Source Control Lead Vlad Zhdanov	● Maintains the Git repository ● Ensures correct branching and merging practices ● Assigns users to review code for incoming pull requests
Documentation Lead Brian Satchell	● Selects the documentation tools to be used ● Ensures consistent formatting and quality of documentation ● Ensures all changes have adequate documentation

7. Key Stakeholder Roles & Responsibilities

Primary Webroot Contacts	
Name/Title	**Email**
Greg Diener	GDiener@webroot.com
Eric Klonowski	EKlonowski@webroot.com
Fred Krenson	FKrenson@webroot.com
Jason Davison	JDavison@webroot.com