A Brief Technical Overview of the Cyberlytic Platform

Dr. Edward Amoroso
Chief Executive Officer, TAG Cyber LLC
Research Professor, NYU Tandon School of Engineering
Adjunct Professor, Stevens Institute of Technology
eamoroso@tag-cyber.com

Version 1.0
January 2018

Abstract

This brief technical note outlines the underlying design and functional operation of the Cyberlytic platform, including how web traffic is analyzed for evidence of cyber security threats, processed using artificial intelligence-based methods, and then assigned risk scoring based on the sophistication, capability, and effectiveness of the observed attack.

Preface

This note is based on on-going interactions between the author and several managing principals from Cyberlytic during the 4Q17 to 1Q18 timeframe. The note incorporates learnings from the author based on review of technical materials, as well as multiple platform briefings from the technical team. It is written to help anyone interested in the Cyberlytic platform better understand its foundational design and operation in reducing enterprise cyber risk for web application traffic.

Introduction

The Cyberlytic Platform is designed to address web application security in a way that avoids the weaknesses of conventional web application firewalls (WAFs). That is, WAFs typically rely on static signature and regular expression matching, with manual procedures required by humans to define and make updates to filtering rules. This traditional approach has proven ineffective at detecting zero-day attacks and polymorphic variants that are designed by capable actors to side-step any predictable rules.

The primary functional component in the Cyberlytic Platform is called the Threat Profiler. This commercially available tool addresses web application risk through machine-learning systems that use automation and intelligent algorithms to dynamically identify threats such as conventional cross-site scripting (XSS) and SQL injection attacks, as well as previously unseen zero-day exploits. The Threat Profiler, which can be application-integrated or stand-alone, also prioritizes cyber attacks based on the measured risk to business assets. Such prioritization improves incident response process flow and supports generation of risk impact displays, alerts, and reports for management.

A second planned component of the Cyberlytic Platform, called the Threat Defender, is currently in Beta testing. The Threat Defender is being designed as an integrated or stand-alone security component that will actively remove malicious content from inbound communication with a protected web application. It will do so, based on information provided by the Threat Profiler, and will interoperate seamlessly with ecosystem components such as existing commercial WAFs.

This paper provides a brief technical overview of how Cyberlytic's unique Threat Profiler approach to web application security is designed and how it utilizes a patented threat classification approach based on machine learning. That threat classification approach is shown to include powerful means for combining deep packet inspection with algorithms that use learning to measure sophistication, capability, and effectiveness of a web application attack.

Cyber Threat Profiler Overview

The Cyberlytic Threat Profiler is typically deployed as a virtual appliance, integrated with an existing SIEM or WAF, and connected to the web application either locally or via cloud infrastructure. It collects all inbound web application traffic, analyzing request and response protocol traffic to identify potential cyber risks. Inbound HTTPS traffic must be subjected to SSL termination for the Cyberlytic solution to review the web application traffic.

The process flow is as follows: Encrypted HTTPS traffic originates from the client web browser through an enterprise firewall to an SSL termination point. Once the SSL has been terminated, the HTTP traffic is mirrored via port scan to a front-end component known as a Connector. This component is also typically deployed as a virtual machine image. The Connector then passes the HTTP stream to the Profiler for analysis.

The Profiler maintains its own connectivity to the web server and applications directly to support the security analysis. The primary goal in the Profiler is to determine a risk score based on relevant factors and using advanced analytics. The output from the Profiler is provided via a customer portal which offers analysts with a real-time view of risk exposure, including threat analysis, targeted hosts, attack methods observed, and risk measured across timeline.

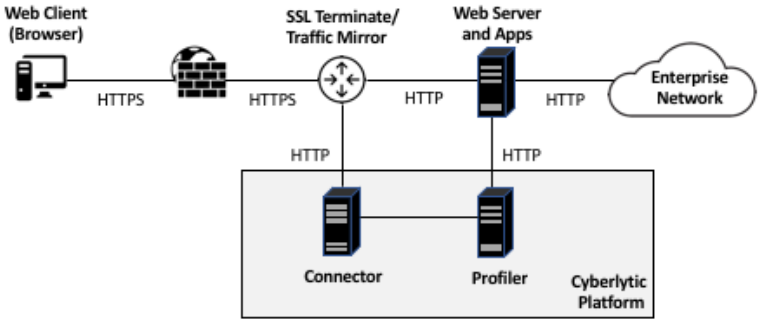

Figure 1. Deployment of the Threat Profiler

The data flow for the Profiler can be viewed in more detail as a path from captured HTTPS traffic to a series of displays, alerts, and reports of any observed malicious traffic. The first step is that browser requests and server responses are passed to an SSL termination point to produce plain text HTTP. This traffic is then captured mirrored via port scan methods to a so-called Connector, which can be a conventional web server agent, lightweight Nginx Proxy, or a standard network connector type.

The role of the Connector is to make the HTTP traffic stream available to the Profiler for analysis, which begins with a broad functional decision point. That is, as an initiated web application session is passed to the Profiler, the first decision that must be made is whether the HTTP activity looks anomalous in any

way. This is done through advanced profiling methods using labeled examples in supervised and semi-supervised machine learning modes. If the traffic is not anomalous, then it is treated normally.

The second decision that must be made for any traffic deemed to be anomalous is whether the activity shows characteristics of maliciousness. If no evidence of maliciousness is detected, then the anomalous traffic is displayed for the security administrator; however, if maliciousness is detected, then security response is required in the form of preventive action or real-time response. This is implemented as displays, alerts, or reports of malicious activity.

The decision making by the Profiler involves the following three web application traffic characteristics that offer hints as to potential risk:

- *Sophistication* – This is a measure of the overall quality and technical soundness of the attack being observed.
- *Capability* – This examines the source of an attack with emphasis on detection of human versus machine origination.
- *Effectiveness* – This measure uses the server response to determine if the attack produced a normal or anomalous condition.

The output of the Profiler is a series of reports, displays, and alerts that correspond to three possible conditions: Traffic reported as being normal, with no evidence of anomalies or malicious content; traffic being reported as anomalous, but with no evidence of malicious content; or traffic being reported as both anomalous and likely to be malicious. In addition to reports and displays, security alerts are generated for malicious traffic detection. SIEM and WAF integration ease this reporting function.

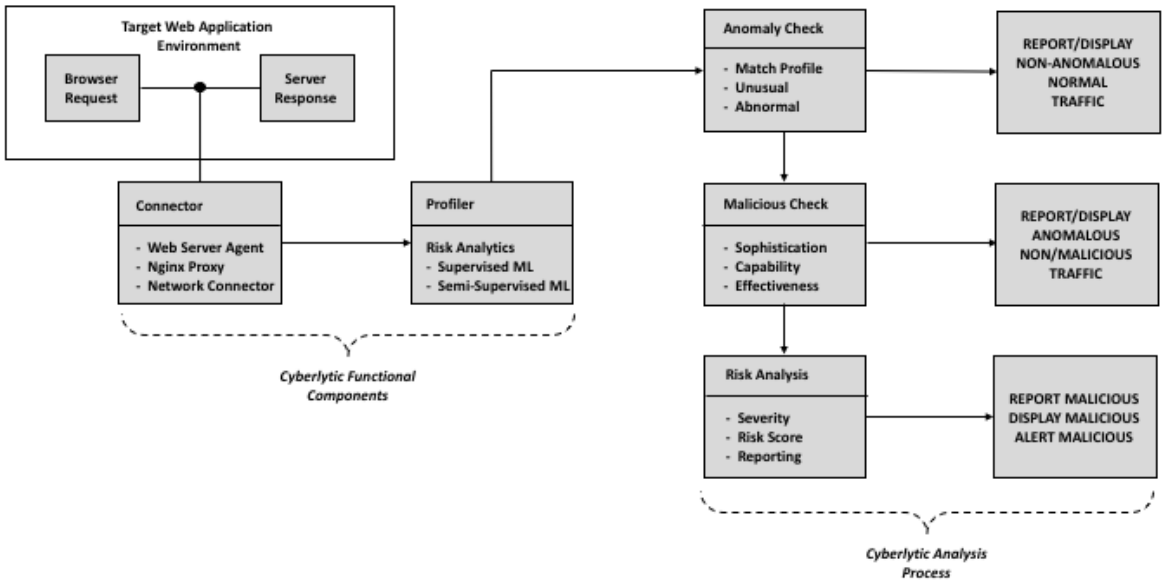

Figure 2. Threat Profiler Decision Flow

In the decision-making process for detection of maliciousness, the Profiler utilizes two modes of machine learning algorithms in its attack detection and operation: *Unsupervised machine learning* to detect web threats, and *semi-supervised learning* to prioritize attacks. Each of these are designed to

detect conventional attacks such as XSS and SQL-injection, as well as dynamic zero-day exploits. Typical training time to minimize false positives involves covering roughly 10,000 web session events.

The unsupervised machine learning analyzes characteristics of data flows to the web application based on profiles of normal behavior. Each web server is treated uniquely since each will exhibit its own profile based on usage trending, time of day, character lengths, and distribution. The Profiler then determines whether requests are derived from normal distribution or if the HTTP traffic is anomalous and should be passed to the classification process for fingerprinting and risk assessment.

The semi-supervised machine learning focuses on SQL injection, XSS, and Bash attacks using the three key characteristics for analysis mentioned earlier – namely, sophistication (attack string quality), capability (human or machine attack origination), and effectiveness (normal or abnormal response from server). These characteristics are normalized through machine learning to determine the risk of each attack. Highly sophisticated attacks from capable humans with abnormal response are thus the highest risk.

Cyber Threat Profiler Design

The overall machine-learning process in the Profiler involves a stepwise method that normalizes ingested web application request and response session information into tokenized representations that can be classified in terms of risk. This overall Profiler operation is best represented – from a design and implementation perspective – in the context of a series of functional primitives that combine in a bottom-up manner to support web exploit detection. Below are brief descriptions of these primitives that collectively support Profiler functionality in the Cyberlytic platform:

Primitive: Data Parsing

The first functional primitive in the Profiler involves ingested session data in JavaScript Open Notation (JSON) which is statistically processed using the R language (developed many years ago at Bell Labs). The purpose of the front-end parsing is to make initial determinations as to whether anomalies exist based on review of URL query parameters, session data, and request headers. The output of this primitive is either a positive (50% or greater confidence) or negative determination of anomalous activity.

Primitive: Send to Classifier

The second functional primitive involves a process of methodically checking for attack strings and types based on a series of preloaded libraries for SQL, XSS, and Bash written in C. Should any of these attack types be detected, the output is sent off to update and improve the classification of malicious data, which is essentially a learning process. Future versions of the platform will introduce more web attack types to be focused upon.

Primitive: Classify

The third functional primitive involves existing training data is combined with identified attack strings or types to develop a risk score based on a fingerprint of the object being reviewed. This involves estimation of effectiveness using IP location, character string analysis, and detection of scripts. The output of this primitive is a fuzzy risk score based on the three main parameters of sophistication, capability, and effectiveness. Once complete, the next object can be ingested for data parsing.

Primitive: Initialize Daemon

This primitive is part of the anomaly daemon administration and set-up (including setting port number 6314 for daemon operation). This primitive involves establishment of workspace, functions, libraries,

and relevant packages. An initialization is performed to define anomaly thresholds and tolerance levels that will be used for profiling and classification. Characteristics are also set and stored in a dataframe for ingested MongoDB connections.

Primitive: Begin Daemon
This simple primitive uses predefined values to determine the interval for ingesting and aggregating data, with an initial value of 60 seconds, plus any processing time required for the data to be set up. This primitive makes a call to the corresponding function to control the anomaly filters and aggregation methods, to call for garbage collection, and to manage daemon operation including sleep time.

Primitive: Get Events
This primitive is focused on checking for normal events using a vector of strings returned via Mongo query. If no normal traffic is identified by the analysis, then additional checking for malicious activity is performed and used as the basis of stored update. Otherwise, request headers are parsed out and recorded in Mongo.

Primitive: Get Traffic Volumes
This primitive is designed to connect traffic volumes to time bins, with caching of timestamps into the Redis open source database. A traffic loop is established that focuses on managing the time synchronization of traffic volumes during analysis. Standardized timestamps are used to avoid the problems of trying to do analysis over unequal time grids. The primitive completes when the volumes of a dataframe have been established.

Primitive: Daily Data
The primitive utilizes a daily data object created in Mongo that contains reporting parameters such as timestamps and traffic volumes for that day. Each day, a null object is first created and then populated for twenty-four hours until a new null object is established. If no malicious activity is present, then the daily data is integrated into Mongo.

Primitive: Characteristic Analysis
This important primitive supports the web session security analysis. This is where a given event is reviewed in the context of data stored in Redis. Analysis addresses character length and character distribution to determine normality. Request headers are investigated, and a tokenization procedures is carried out to model the captured traffic. Repeat patterns observed in traffic are identified and extracted for subsequent analysis (e.g., a traffic stream of ‘abcabcabcabc’ produces ‘abc’). Once the analysis is completed, a normalized risk score between 0 and 1 is calculated.

Primitive: Store Response Bodies
This final primitive parses out responses for classification and helps manage the ongoing execution of the daemon for subsequent incoming web application session traffic for analysis by the Profiler.

Cyber Threat Defender Planned Operation
The Cyberlytic team is currently developing a corresponding Defender component that includes active mitigation of web application threats. The architectural objective for the Defender is to involve hybrid operation where the component can reside in the web traffic flow, as one might find with a web application firewall, but also with direct API connectivity to existing mitigation tools from complementary cyber security vendors. Some practical considerations include the following:

- *Active Versus Passive Operation* – When operating off-line in passive mode, the Cyberlytic platform can be easily integrated into an enterprise architecture without due consideration for service interruption – which is an advantage found in any passive cyber security tool. Once operating in active mode, however, the Defender must incorporate enterprise-grade (or even carrier-grade) functional assurance of continued fail-safe operation.
- *Integration with Third-Party Components* – Establishing connectivity with external vendor WAFs and other security tools is a natural design objective for the Cyberlytic platform. Determination of which tools require such integration (e.g., Cisco, F5, Palo Alto Networks) is an on-going task driven by market interactions with Cyberlytic customers.
- *Human Versus Machine Attack Origination Detection* – Considerable attention is required in any web application security tool to differentiate between human and machine-originated attacks. The Cyberlytic platform comfortably deals with both cases, but any design decision that involves actions such as source IP shuns would require different handling if the origin is determined to be a botnet.

As of the date of this report writing (1Q2018), the Cyberlytic team is currently in beta testing with the Defender and hopes to have an active mitigation platform available for its customers in 3Q2018.

References

Cyberlytic Intelligence Web Security – Fact Sheet, https://www.cyberlytic.com/
Cyberlytic Profiler Analytics – https://www.cyberlytic.com/
Private Notes from Cyberlytic Development Team, 4Q2017
The Profiler – AI For Web Security – Technical Data Sheet, https://www.cyberlytic.com/