How to Trick the Borg

Threat Models against Manual and Automated Techniques for Detecting Network Attacks

Carl Sabottke[a], Daniel Chen[b], Lucas Layman[c], and Tudor Dumitraș[d]

[a] Carl Sabbotke was a graduate student at the University of Maryland Institute for Advanced Computer Studies, 2126 AVW Building, College Park, MD 20742, USA. E-mail: cfs121090@gmail.com

[b] Daniel Chen was an undergraduate student at the University of Maryland Institute for Advanced Computer Studies, 2126 AVW Building, College Park, MD, USA, 20742. E-mail: dchen1395@gmail.com

[c] **(Corresponding)** Lucas Layman was with the Fraunhofer Center for Experimental Software Engineering, 5825 University Research Ct, Suite 1300, College Park, MD 20740, USA. He is currently Assistant Professor at the University of North Carolina at Wilmington, CIS Building, 601 S. College Rd., Wilmington, NC 28403, USA. E-mail: laymanl@uncw.edu.

[d] Tudor Dumitraș is Assistant Professor at the University of Maryland Institute for Advanced Computer Studies, 2126 AVW Building, College Park, MD 20742, USA. E-mail: tdumitra@umiacs.umd.edu

Abstract

Cyber attackers constantly craft new attacks previously unknown to the security community. There are two approaches for detecting such attacks: (1) employing human analysts who can observe the data and identify anomalies that correspond to malicious intent; and (2) utilizing unsupervised automated techniques, such as clustering, that do not rely on ground truth. We conduct a security analysis of the two approaches, utilizing attacks against a real-world website. Through two experiments—a user study with 65 security analysts and an experimental analysis of attack discovery using DBSCAN clustering—we compare the strategies and features employed by human analysts and clustering system for detecting attacks. Building on these observations, we propose threat models for the human analysis process and for the unsupervised techniques when operating in adversarial settings. Based on our analysis, we propose and evaluate two attacks against the DBSCAN clustering algorithm and a defense. Finally, we discuss the implications of our insights for hybrid systems that utilize the strengths of automation and of human analysis to complement their respective weaknesses.

Keywords: cyber attack; human factors; unsupervised learning; DBSCAN; log analysis

1 Introduction

Because of the ongoing cyber arms race, the security community must keep watch for new, previously unknown attacks. For example, attackers seeking to compromise Web servers may try to exploit new vulnerabilities in Web frameworks, to target a wide array of potential misconfigurations, or to obfuscate the request parameters used for delivering the attack. The incentive for creating such new attacks is to be able to compromise Internet servers while remaining undetected.

In practice, attackers face defensive systems that, like the Borg from Star Trek, are part human and part machine. The human part corresponds to *security analysts* who scrutinize the server logs and identify anomalies that correspond to malicious intent. For example, many security firms provide managed security services (MSS) that use analysts trained to monitor network activity to generate and investigate alerts and to respond to incidents. The machine part consists of *supervised* and *unsupervised* data analysis. Supervised techniques assist human analysts by blocking or flagging potentially malicious activity against a wall of network traffic by using malware signatures and other ground truths of known attacks. *Unsupervised* techniques include anomaly detection [?], which aims to detect outliers in the data distributions, and clustering [?, ?, ?], which aims to group similar sessions in the hope of separating benign and malicious activity.[1]

In this paper, we explore how an attacker might exploit weaknesses in human analysts and unsupervised techniques to launch an undetected attack. Unsupervised learning systems excel at processing large quantities of information and at identifying new correlations among features of the data set, but may produce an unacceptable number of false alarms. Conversely, human analysts accurately discern existing or new forms of malicious activity, but are unable to cope with large amounts of information. The challenge for an attacker is to select a strategy to bypass both types of defenses.

We conduct a security analysis of human and machine approaches for detecting known and potentially unknown attacks. Many attack detection systems have, in practice, both a human and a machine component, as security analysts often utilize various tools to process the raw data. While human analysis and automated tools serve complementary purposes, the effectiveness of the combined system—and the options that an adversary has to evade it—are not well understood. It is challenging to design experiments that capture both the effectiveness of human intuition in detecting new attacks and the impact of intelligent adversaries who interfere with the detection mechanism. As a first step in this direction, our goal is to investigate concrete attacks against the human and machine components, to highlight the opportunities for combining them, and to propose a threat model for systems that combine human analysis with unsupervised learning techniques.

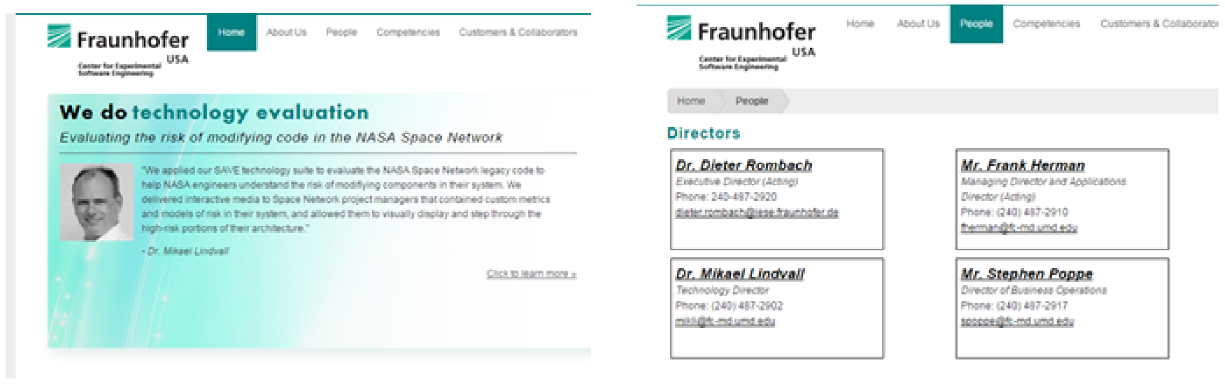
We consider a security threat to be previously unknown if the analyst is not familiar with the threat or if the designer of the unsupervised learning system had not considered it when tuning the system. We utilize attacks against a real-world website (`http://www.cese.fraunhofer.org`) as a vehicle for exploring strengths and limitations of the two approaches and the opportunities for combining them. We analyze the features that can be extracted from the raw Web logs, and we compare the most useful features for human analysts and for an automated clustering system. We also explore adversarial attacks against human analysts and unsupervised machine learning.

The remainder of the paper is organized as follows. We outline our methods and data sources in §2. By analyzing data from a user study, we describe the process by which human analysts discriminate between malicious and benign activity and how this process can be exploited by attackers in §3. In §4, we discuss the feature selection, feature representation, and hyperparameter tuning process of an automated clustering system based on the DBSCAN clustering algorithm [?] that uses features drawn from human detection processes. We also assess the effectiveness of this attack detector in the presence of adversarial interference. Building on the insights from this investigation, we present our threat models in §5. Finally, in §6 we provide guidelines for combining human analysis and automation effectively, and we identify areas where further research is needed.

We make five contributions in this paper:

- We describe a general process that human analysts follow to detect network attacks. While applying this process to our data set, we discovered and reported a new vulnerability in the Database API of Drupal 7 (CVE-2015-6659).
- We compare the features that are most helpful for humans and for an automated clustering technique.
- We propose two attacks against a DBSCAN-based clustering system: the *bridge attack*, which extends an attack previously demonstrated against hierarchical clustering, and the *net attack*, which assumes a weaker adversary. We propose a defense against such attacks and evaluate the effectiveness of these attacks and defenses through simulation.
- We propose threat models for the human analysis process and for unsupervised learning in adversarial settings.

[1]Supervised learning and signature-based automated analyses are essential to effective defense as well, but they are not discussed here as we focus our discussion on previously-unseen attacks.


Fraunhofer USA
Center for Experimental Software Engineering
Home
About Us
People
Competencies
Customers & Collaborators
Home
People

We do technology evaluation

Evaluating the risk of modifying code in the NASA Space Network

"We applied our SAVE technology suite to evaluate the NASA Space Network legacy code to help NASA engineers understand the risk of modifying components in their system. We delivered interactive media to Space Network project managers that contained custom metrics and models of risk in their system, and allowed them to visually display and step through the high-risk portions of their architecture."

- Dr. Mikael Lindvall

Click to learn more »

Directors

Dr. Dieter Rombach
Executive Director (Acting)
Phone: 240-487-2920
dieter.rombach@iese.fraunhofer.de

Mr. Frank Herman
Managing Director and Applications Director (Acting)
Phone: (240) 487-2910
fherman@fc-md.umd.edu

Dr. Mikael Lindvall
Technology Director
Phone: (240) 487-2902
mikli@fc-md.umd.edu

Mr. Stephen Poppe
Director of Business Operations
Phone: (240) 487-2917
spoppe@fc-md.umd.edu

Figure 1: The home page and People page for the Fraunhofer CESE website.

- We provide concrete guidelines for combining human analysis and automation effectively.

2 Data Sets and Methods

Web Log Data. We collect approximately 3 months of Web server logs from the Fraunhofer Center for Experimental Software Engineering (CESE) (`http://www.cese.fraunhofer.org`) website. Fraunhofer CESE is a nonprofit applied research organization in College Park, Maryland, USA with customers in the aerospace, automotive, and medical device domains. At the time of this experiment, the Fraunhofer CESE website consisted primarily of static content, managed by the Drupal Content Management System (CMS) [?] and hosted on an IIS 7 Web server. The website comprises approximately 150 pages serving information about the center, including news items, staff profiles, training offerings, publications, and more. Screenshots for the home page and the People page for this website are shown in Figure 1. Google Analytics reports approximately 4,000 page views per month with 90% of visitors from North America. We analyze logs collected between 12 December 2014 and 11 March 2015, using the server's IIS Advanced Logging Module. The log fields include: date, time, client IP, server port, method, status, URI stem, URI query, the client user agent, the client referrer, and the cookie if applicable. For our unsupervised clustering analysis, we process the log files by grouping requests into sessions. Each session consists of the consecutive requests from the same client IP that do not occur more than 5 minutes apart from each other. We record 146,851 sessions from 21,955 distinct client IP addresses.

Typical attacks observed against the site include repeated login attempts, likely part of a password guessing attack, and attempts to exploit common Drupal misconfigurations, e.g. probing for an admin panel with open access and attempts to insert users or nodes into the site's database. Among the login attempts, we observe 14 sessions with more than 20 login attempts, while 195 sessions (81%) make only 2 login attempts, perhaps in an effort to keep the number of login attempts below a threshold that would raise alarms. Similarly, for admin panel probing, only 16 of the 1,619 sessions make more than 20 requests with"admin" in either the stem or query. Additionally, 225 of the 241 sessions making login attempts (93%) generate HTTP 404 error codes (Not Found) rather than 403 error codes (Forbidden), because these login attempts are targeted against other content management systems, such as Wordpress. Less frequently, we observe attacks against Drupal plugins and extensions with known exploits such as Fckeditor and Uploadify. We do not observe targeted attacks against the Fraunhofer site.

Human Analyst Experiments. In previous work that was foundational to our current analysis, we conducted two controlled experiments with human analysts, both novice and expert, to understand the value of various log fields for detecting attacks and the cognitive processes of security analysts [?, ?]. The first observational study involved 14 self-selected subjects from the Fraunhofer CESE and the University of Maryland computer science department. The follow-up study was conducted online using Amazon's Mechanical Turk. The 65 subjects' self-reported job roles were: 11 IT/network admins, 21 software engineers, 7 researchers, 18 students, and 8 who fell into the "Other" category. The professional (i.e. non-student) subjects reported a range of experiences in their respective fields, with 41% having 1-5 years, 23% having 5-10 years, and 31% having 10+ years of experience. All participants were required to pass a preliminary test that demonstrated basic competence needed to read and understand webserver log file content [**?**].

The 65 participants were shown 50-line excerpts from 6 log files from the Fraunhofer CESE webserver presented in random order: four of the log files contained lines exhibiting malicious behavior (admin login attempts, JavaScript injection, SQL injection, and shell access attempts) while the remaining two files contained benign behavior. The log files used in this experiment were distinct from the log file set later used in our DBSCAN analysis. These website log entries contained the date, time, HTTP method, client IP, client hostname, request protocol, response status, size of the response in bytes, the requested page, and query parameters for each request. For each log file, participants were asked:

1. Does the log file contain malicious activity or not?

   [Definitely malicious—Probably malicious—Undecided—Probably normal—Definitely normal]

2. Why do you think this log file contains malicious activity or not?

3. What were the three most useful log file fields in making your determination?

After reviewing all the log files, participants were asked:

4. Are there any key

   words or patterns that you look for when investigating an attack?

5. Describe in your own terms your process for analyzing the log files? What are the steps for you?

We applied qualitative coding and statistical analysis to the subjects' answers (full details in [**?**] and [**?**]). The analyses identified several features that the analysts used to determine whether a log file contained malicious activity (§3).

Automated Analysis Techniques. Whereas the human experiment involved a forensic analysis of 6 log files, we performed clustering experiments on the entire corpus of the 3-month web log data using density-based spatial clustering of applications with noise (DBSCAN) [**?**]. DBSCAN produces several *clusters* and labels some samples, which do not lie within any cluster, as *noise*. Potentially malicious sessions lie either in the DBSCAN noise or in the territory of a malicious activity cluster. If samples labeled as noise in the DBSCAN clustering are intended to be analyzed as potential threats by human analysts, then it is consequently important to parametrize the DBSCAN clustering in order to reduce the number of noise samples and avoid overstimulation of the human analysts [**?**, **?**].

We elected to use DBSCAN for this work specifically because the notion of noise samples that can be investigated in isolation complements our interest in looking at hybridized approaches that combine machine learning with manual analysis. DBSCAN is also useful mathematically in that it can generate clusters of arbitrary shapes and does not require a priori knowledge of the number of clusters. Even though cluster number is not specified explicitly prior to analysis, DBSCAN involves two tuning hyperparameters which impact the number of clusters and noise samples: the minimum number of samples in a cluster and ϵ, the maximum distance between two samples for them to be considered as part of the same neighborhood. Two samples separated by more than ϵ can be in the same cluster if they are connected by other samples within ϵ in the feature space. Consequently, a large ϵ allows for the formation of larger clusters, while reducing ϵ typically leads to smaller clusters and an increased cluster count.

To evaluate clustering performance, we calculate the silhouette score [**?**] as an internal validation metric. Here, the silhouette score is assessed for each point in each cluster by calculating the average distance between the point and every other point in the assigned cluster. This value is subtracted from the lowest possible value of average distance between the point and every member of another cluster. Assuming that the average distance to every point in the closest by average neighboring cluster is less than the average distance to every point within the assigned cluster, the silhouette score for each point is is arrived at by dividing the average distance difference by the average distance to the neighboring cluster's points. The reported silhouette score is then the average of the silhouette score for all points in the clustering, and we use this silhouette score to assess the mathematical appropriateness of our clustering. Intuitively, a high silhouette score indicates that the clusters are compact (each point is similar to the other points in its cluster) and well separated (each point is unlike the points in other clusters). As a form of external validation, we manually examined whether those clusters corresponded to keywords associated with attacks (i.e. "login," "admin," and database insertion keywords). These keywords were chosen based on feedback from the human experiment. Finally, we manually analyze the clusters to identify normal and attack behaviors.

When simulating an adversarial environment, we assess the impact of attacks using the adjusted Rand index to compare the similarity between the clusters produced in the absence and in the presence of adversarial interference. The Rand index ranges in value from 0 to 1 where a value of 1 denotes maximal similarity between two clusterings. The adjusted Rand index corrects for random chance-related overlap between clusterings and retains a maximum value of 1 but is allowed to have negative values when the unadjusted Rand index is below the expected value associated with random clustering.

Threats to Validity. The limitations of the human analyst experiment are more fully described in [**?**]. The Web attacks observed, and the features used to detect them, may be specific to the Fraunhofer CESE site. In particular, the site has a small user base, relatively small traffic volume, no third party advertisements, no payment processing or customer database, and no subscription content. We therefore caution the reader against generalizing our observations about the features that are most effective for detecting attacks. Instead, our goal is to emphasize the process of extracting features used by human analysts and for clustering and the reasons why the feature sets may be different in the two approaches. Additionally, our clustering attacks and defenses target specifically the DBSCAN algorithm, yet we believe that the threat models we propose are generally applicable. Moreover, as we were able to extend an attack against agglomerative clustering, we expect that other researchers will build on our DBSCAN attacks and defenses for performing unsupervised learning in adversarial settings.

3 Human Techniques for Detecting Attacks

3.1 Features Utilized by Human Analysts

We consider the features used by human analysts to identify malicious activity to be the set of log file fields the humans examined most-frequently and the patterns they searched for in these fields in [**?**, **?**]. We examined: a) did the analyst correctly identify the log file as containing malicious activity or not (Question 1 from the previous section), b) which log fields did they examine and what did they look for (Questions 2-5), and c) the strategies used by strong- vs. weak-performing analysts.[2] The features below are presented in order of the number of times the participants used the features to successfully categorize malicious vs. non-malicious log files.

1. *Keywords Requested Page content* - The requested page is the most direct indicator of a requester's intentions. Analysts would look for keywords indicative of common attacks, such as SQL command keywords, Javascript code, terms such as "admin" or "password", and access to known backdoor shells. The Requested Page field was identified as the most useful field in 65.4% of log file assessments performed by the analysts, which is the highest percentage by a wide margin.

2. *Request origination in Hostname and Client IP* - These fields are used in conjunction with the requested page to make an assessment. The Client IP and Hostname are useful for determining if a request is from a malicious or innocuous source. Some requests originated from blatantly malicious hostnames, e.g, `*.blackhat.hu`. Client IPs can also be examined against IP reputation services. Further, client requests originating from countries that have no business ties with [Redacted] may be suspicious.

3. *Failed requests in the Status Code* - The Status code identifies failed requests, which may be linked to malicious resources. For instance, requesting access to an administrator panel that is either non-existent or locked to the user will result in a HTTP 404 or 403 response.

4. *HTTP method* - The presence of POST methods can be useful for identifying certain classes of attack, particularly when the website does not provide many user input options. For example, the Fraunhofer

[2] 24% of subjects correctly assessed all 6 log files, 28% were correct on 5/6, 29% were correct on 4/6, 14% were correct on 3/6, and 5% were correct on 1/6. See §2.6 of [**?**] for details.

CESE website serves mostly static content, with the only legitimate POST requests supported by a simple search bar and by user logins.

5. *Requested page and query parameter length* - The number of printed characters in the request string was sometimes an indicator of suspicious activity. Analysts noted that this often accompanies an injection attack on an input field. Further, some of the more lengthy request strings were due to URL percent encoding meant to mask a Javascript injection (e.g. Log E in [**?**]).

These features are derived from accurate (correct) log file assessments. We observed that incorrect log file assessments used other features, and that the feature set used by a participant was a function of their years of experience and self-identified expertise with computer security and web platforms. A detailed discussion of the features used by strong- vs. weak-performing analysts may be found in [**?**]. The features used in unsuccessful assessments are incorporated into threat models for human attackers in §5.1. in many cases,

3.2 Human Analysis Process

Patterns emerged in the workflow of human analysts. Consider one log file where an attacker repeatedly tried accessing an administrator panel. Some analysts began by visually scanning the log file, while others sorted the file (presented in an Excel workbook) according to fields such as requester IP and status code. They then visually scanned the log file for the features in §3.1, such as interesting keywords in the requested page (e.g., "admin" and "login") or failed requests (status 40X). When a failed login attempt was found, the analysts often looked at other log entries from the same client IP and other features within the log entry to verify that the suspicious activity was in fact malicious. Manual log analysis is an instance of *information foraging* [**?**] – the processes by which humans allocate time and resources, identify relevant environmental queues, and select and pursue information in an environment where the information available adapts over time.

We conducted a qualitative coding analysis, described in more detail in [**?**], to understand the strategies (i.e., the algorithms) by which human analysts combined and applied these features from §3.1. The strategies for using the features fell into three categories, which when combined form an overall analysis process exhibited by the study participants:

1. **Grouping or filtering log entries** - The grouping strategies are a means to make the searching process easier. Some subjects grouped the log entries by Client IP or Hostname to track a single user's behavior over time, whereas other subjects filtered out spiders and search bots that generated many log entries but are generally harmless.

2. **Search the log file** - Searching strategies are those by which the analyst visually scans (perhaps with find/replace assistance) for suspicious log entries. Searching involves the direct comparison of a log entry to one of the features in §3.1. This approach is susceptible to "anchoring" [**?**], wherein initial estimates of the desired finding incorrectly dominate the results.

3. **Correlate / confirm hypotheses** - Correlation strategies verify whether suspicious activity is malicious or not by combining information from multiple log fields. For example, analysts examined a series of requests from a single IP to determine if the pattern is unusual given the rest of the traffic on the webserver. If numerous requests are received in a short amount of time, this is not a cause for concern if the client is requesting webpages one would expect on the server or if the client is a search crawler. However, if there are numerous requests for requested pages containing keywords such as "admin" or "password", this may be indicative of a client maliciously scanning for a backdoor. Further, a 404 or 403 response does not necessarily indicate malicious intent: it could be a crawler with an out of date index.

This process becomes ineffective when the human analysts must process large amounts of data (§5.1). However, even when employing data analysis tools for pre-processing the data, an experienced human analyst is often needed to confirm whether unusual behavior corresponds to new attack patterns.

3.3 Vulnerability Discovery

The third step in the analysis process described above relies on human intuition. While it is difficult to measure the effectiveness of intuition systematically, we have confirmed that the process can yield new discoveries. We applied the human analysis process, derived from our user study, to our complete data set of Web server logs (described in §2). The aim of this manual analysis was to identify and categorize the recorded attacks. We observed that attacks involved SQL injection and were not adequately described in existing references, thus, we manually reviewed the Drupal source code to gain a better understanding of these threats. During this code review, we discovered a *new SQL injection vulnerability* in the Database API of Drupal 7 (CVE-2015-6659, CVSS v2 Base Score: 7.5, Exploitability Subscore: 10.0). This vulnerability was included in DRUPAL-SA-CORE-2015-003 and was patched in Drupal version 7.39. Due to configuration requirements needed for implementing the exploit, it is unlikely that an attack against this vulnerability will be observed in the wild.

4 Clustering-Based Attack Detection

While human analysts can be effective at detecting attacks, including previously unseen attacks, the scale and speed of information in modern networks is too much for humans to process. *Supervised* and *unsupervised* automated analyses can help tame this complexity. In this section, we focus on an *unsupervised* technique that can help detect attacks that do not match a preprogrammed signature or traffic filter.

Unsupervised machine learning can discover unexpected connections between attacks and various features derived from the raw data. Additionally, these techniques can separate anomalous, but benign, activity (e.g. 403 errors related to misconfigurations) from attacks (e.g. 403 errors resulting from admin panel probing). Nevertheless, the choices made during the design of a clustering-based attack detector have a substantial impact on the effectiveness of the detector. In particular, the features selected for clustering the data play a major role. In this section, we review the design process for such a detector, with the aim of *comparing the features that are most helpful for humans and for the clustering technique.* We also assess the impact of these features and design choices on the ability to detect new attacks.

4.1 Features and Design Choices

A clustering algorithm groups sessions together according to how similar they are to each other. To compute the similarity between, the algorithm relies on a *distance measure*, which takes into account several features of the data. It is difficult to determine *a priori* which features would be useful to detect new attacks; in practice, security analysts propose multiple candidate features during the design stage of an automated detector and then assess their effectiveness. We construct an initial set of features that are potentially useful for detecting unknown attacks guided by insights from the human analyst experiment from Section 3 and more fully reported in [?]. The three months of web log data were preprocessed into units of analysis called *activity sessions.* An activity session is a set of log entries than begins when a request is received from a new IP address. An activity session terminates when no log entries from the IP address were recorded for 5 minutes or more. Subsequent requests from a previously-seen IP address begin a new activity session if the interval is 5 minutes or more. The initial feature set for unsupervised learning based on the activity sessions is shown in Table 1 along with intuitions related to the potential utility of each feature. The initial features fall into three main categories: (1) HTTP request stem and query content, (2) traffic timing patterns, and (3) HTTP method and status features.

We also choose the DBSCAN clustering algorithm [?] for our detector. An advantage of DBSCAN over other clustering techniques (e.g. k-means or agglomerative clustering) for attack detection is that it identifies dense clusters and it labels the points that do not lie in any cluster as noise. The resulting clusters correspond to regular activities—which could be malicious if the site is frequently under attack—and the noise sessions highlight suspicious events. However, using a large feature set with DBSCAN makes it less likely that two sessions will appear similar, and this could result in many clusters and many noise sessions. Because these sessions require manual investigation, we must select features and parameterize DBSCAN in a manner that avoids creating a large number of clusters and of noise samples.

Feature	Description	Intuition
POST Requests	Number and percentage of POST requests in the session	A large number of POST requests could be indicative of injection attacks.
Keywords	Individual and combined counts for keywords in request parameters. Included: “script”, “select”, “#”, “;”, “%”, “..”,“admin”,“login”, “Error.aspx”, “cgi-bin”, “includes” , “misc”, “modules”, “profiles”, “scripts”, “themes”, “People”, “robots.txt”, “highslide.js”, “mootools.js”, “fckeditor”, “uploadify”, “node/add”, “user/register”, “CHANGELOG”, “user/password”, “user/login”, “filter/tips”, “comment/reply”	Used to detect SQL injections, hex encoding, login attempts, admin panel probing, etc.
Tor*	If the user is behind a Tor proxy	User is attempting to mask their identity
Geography*	The ranking of the country on http://globalsecuritymap.com/, an ordered list of how much cybercrime occurs in countries	Suspicious when requests are from countries with high ranking or ones that do not normally send requests
Number of Connection Spikes	A connection spike is defined as a one-minute window when the client sends more than 60 requests	Connection spikes can be indicative of probing for vulnerable modules or DOS attempt
Longest Connection Burst	A connection burst is a period of time where every minute is a connection spike, but the threshold is lowered to 45 requests/minute	Same reason as connection spike feature
Unsuccessful Requests	The number and percentage of requests that do not return status of 200, 301, or 304 and specifically the numbers of requests that return 401, 403, and 404 status codes	A large number of 403 or 404 statuses can be indicative of probing for vulnerable modules
Workday	If the session begins during the workday (8:00 AM to 6:00 PM EST)	It is possible that malicious attacks would occur more frequently outside of the workday
New Pages Accessed*	The number and fraction of pages that had not been accessed before. There wasa training period of 3 weeks in order to collect a preliminary dataset of requested pages	Infrequently visited pages could be indicative of anomalous behavior

Table 1: Initial feature selection and intuition.

Feature	login	admin	DB	LOG.txt	robots	People	uploadify	fck	cgi-bin	%	Spikes	Bursts	POST	403	404
login	241	208	10	0	17	117	2	2	2	2	19	11	27	42	225
admin	208	1619	2	0	17	119	16	15	2	2	23	11	13	75	248
DB Add	10	2	238	2	2	0	0	0	0	0	0	0	10	234	4
LOG.txt	0	0	2	9	2	0	0	0	0	0	0	0	0	2	2
robots.txt	17	17	2	2	6208	1449	0	0	0	37	73	78	0	187	590
People	117	119	0	0	1449	8216	2	2	0	186	156	241	32	54	576
uploadify	2	16	0	0	0	2	16	2	0	0	2	0	0	16	16
fckeditor	2	15	0	0	0	2	2	27	0	0	4	0	0	21	27
cgi-bin	2	2	0	0	0	0	0	0	5	5	2	0	5	0	5
%	2	2	0	0	37	186	0	0	5	382	10	0	5	13	108
Spikes	19	23	0	0	73	156	2	4	2	10	274	87	19	107	70
Bursts	11	11	0	0	78	241	0	0	0	0	87	322	23	65	37
POST	27	13	10	0	0	32	0	0	5	5	19	23	1339	22	1243
Status 403	42	75	234	2	187	54	16	21	0	13	107	65	22	1227	115
Status 404	225	248	4	2	590	576	16	27	5	108	70	37	1243	115	7581

Table 2: Session Feature Overlap. Diagonal entries indicate the total count for each feature.

Feature selection. Table 2 shows the intersection of sample candidate features within sessions and highlights relationships between these candidate features and attack patterns. For example almost 50% of sessions attempting logins also visit the site's People page, presumably either to bolster the dictionary used in the subsequent login attempts or to deceive human analysts reviewing log files. Additionally, 225 of the 241 sessions making login attempts (93%) generate 404 error codes instead of 403 error codes indicating that these login attempts are targeted against, for example, Wordpress, rather than the Drupal CMS that the web server is actually running. Another notable indication in Table 2 is that the incidences of traffic spikes and bursts tend to have relatively low overlap with login attempts and admin panel probing. Only 23 of the 274 sessions with traffic activity spikes also involve either login attempts or admin panel probing, while 73 of the 274 sessions with activity spikes access robots.txt. Consequently, placing a high emphasis on these traffic timing based features can often increase the number of false positives among groups of suspicious and malicious activity. Table 1 then shows our full initial feature set, along with intuitions related to the potential utility of each feature.

To avoid producing too many clusters and noise samples, we must further reduce this initial set by selecting the best features for separating benign and attack sessions. Moreover, we must consider that the attack sessions are produced by an adversary who wishes to defeat the attack detector. Filter and wrapper methods are automated techniques to select features for clustering by maximizing internal validation metrics (e.g. the silhouette score [?]), but they may result in clusters that do not discriminate between benign and malicious activities. LASSO feature selection has also been shown to be insecure in adversarial settings [?]. Hence, we adopt a heuristic approach that emphasizes the robustness to adversarial manipulation for feature selection.

We start by removing the features marked with an asterisk in Table 1. An attacker can easily poison the distribution of the New Pages Accessed feature by crawling the site and then issuing requests that result in a large number of noise samples. The Tor feature corresponds to very few samples in our data set. The Geography feature, while useful for the human analysts (see §3.1), is not a good fit for clustering because it results in a poor feature space geometry, as the ranking of countries does not provide a good separation between the likely sources of attacks and of normal requests. While it may be possible to assign weights to better correlate feature values with attacks, such manual tuning would not be robust to changes in the cyber threat landscape. After this step, we are left with a set of 29 features.

We then create a second feature set by further pruning the features and only include features that are causally related to attacks or at the very least burdensome for an attacker to manipulate. This results in the removal of Workday, features related to status codes other than 403 and 404, the percentage of POST requests and most keywords. For example, keywords such as "fckeditor" and "uploadify" suggest specific attacks and are not helpful for detecting previously unknown attacks. We give secondary consideration to the silhouette score and the cluster purity with respect to chosen keyword features, which are causally related to attacks (i.e. "login," "admin," and database insertion keywords). This results in a set with 8 features.

Feature representation. After selecting the features, we must also decide their representation. For

		Clusters																
Category	Noise	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Status 403	464	8	2	0	0	383	128	2	0	0	227	0	0	0	0	0	13	0
Status 404	354	2135	3688	626	0	29	0	31	0	107	17	371	4	35	0	61	49	74
Admin	152	0	0	0	603	0	0	0	741	0	0	0	0	0	0	0	49	74
Login	118	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	49	74
DB Add	110	0	0	0	0	0	128	0	0	0	0	0	0	0	0	0	0	0
Misconfiguration	208	11	0	0	0	113	0	2	0	0	40	0	0	0	0	0	0	0
robots.txt	368	2293	3374	0	0	56	0	8	0	0	68	0	4	21	3	0	13	0
Total	1191	49089	92704	626	603	408	128	125	741	107	238	371	78	213	45	61	49	74

Table 3: DBSCAN Cluster Counts for Expanded Feature Set.

example, a feature related to the number of POST methods in requests can either be represented as binary (whether or not the session contains a POST method in a request), an integer (the total number of POST methods in a session), or a rational number (the fraction of POST methods in a session with the session count as the numerator and the maximum count observed in any session as the denominator). Each representation offers different granularity and scaling with respect to this feature, and each representation also places different constraints on an adversary's possible actions. For example, binary features hinder adversarial bridging attacks, but using a large number of binary instead of numerical features also reduces the amount of information contained within the feature space. Additionally, while numerical features are susceptible to adversarial bridging attacks, using a large number of numerical features, each with a wide range, can limit the feasibility and scalability of such AML attacks. We discuss the adversarial impact of feature representation in more detail in Section 5.2. Unless otherwise noted, in our analysis we scale numerical features between 0–1, to avoid giving too much weight to a specific feature, and we represent them as fractional values.

Hyperparameter tuning. Because each scaled feature takes values between 0 and 1, we set $\epsilon = 0.1$ to balance cluster number and cluster size. We first perform DBSCAN clustering on a subset of 25,000 activity sessions, and for each resulting cluster we identify the *core points*, which have at least 10 points within an ϵ radius (a cluster may also include non-core points, which are nevertheless reachable from other points in the cluster). Because the core points are representative for the cluster, we inspect them manually to determine the type of activity that each cluster corresponds to. We then use these cluster, and their core points, to label the remaining activity sessions in the dataset. Initially performing the clustering on a subset of the data, rather than all 146,851 activity sessions, allows us to tune the activity labeling and to simulate a practical clustering analysis scenario in the which clustering is not redone each time more data is introduced to the server. Instead, an initial preliminary clustering is established and then additional data is fit into this pre-existing set of clusterings. Data points that do not fit into any pre-existing clusters are then labeled as noise by the DBSCAN algorithm.

4.2 Automated Detection Results

Tables 3 and 4 illustrate the results of DBSCAN clustering with the 29-feature and 8-feature sets, respectively. The tables show the number sessions in each cluster according to various categories of activity. The clustering with 29 features achieves a silhouette score of 0.88 with 17 clusters and produces over 1,000 noise samples along with two large normal activity clusters (Clusters 1 and 2).

The clustering in Table 4 uses the reduced feature set, involving only binary variables for "admin," "login," and database insertion keywords (i.e. "user/register" and "node/add") along with numerical variables for 403 statuses, 404 statuses, POST methods, connection spikes and connection bursts. This results in 7 clusters and 235 noise samples. Cluster 4 represents sessions that combine login attempts with admin panel probing attacks. In the expanded feature space, these 183 attack sessions are split between clusters 16, 17, and the noise. Similarly, cluster 2 in Table 4 represents admin panel probing, while these attacks lie in clusters 4, 8, and the noise in Table 3. Cluster 6 for both feature sets represents database insertion attempts, yet, for the 29 dimensional feature space, 100 more of the samples are labeled as noise. Additionally, a large number of server misconfiguration errors are clustered as normal activity in the 8 dimensional feature space, while they are separated into different clusters in the 29 dimensional feature space. The coarser granularity of the results for the reduced feature set suggests that some information is lost. Nevertheless, this clustering achieves a silhouette score of 0.99, which indicates that the clusters are compact and well separated. In consequence, the adversary cannot assume that the attacks will remain stealthy—even if they do not match any known signatures employed by firewall and intrusion detection systems—because an automated clustering algorithm is able to separate the attacks from normal activity, bringing the threat to the attention of the site's administrators. The attacker would then have to take steps to evade such an automated system.

4.3 Attack Detection with Adversarial Interference

Our clustering-based detector will help analysts discover previously unknown attacks only if these attacks form new clusters or are singled out as noise data points. However, the adversary may inject additional sessions, which resemble both the attack and the regular traffic, to merge the attack sessions into a larger cluster, characterized by benign core points. To assess the adversary's ability to prevent such discoveries, and the impact of our design choices on the adversary's success odds, we consider cluster poisoning attacks against our detector. We adapt a *bridge attack*, previously described against agglomerative clustering [?,?,?], to DBSCAN, we introduce a new *net attack* and we evaluate a new defense against these attacks.

We adopt a formalism, proposed in the prior work on clustering in adversarial settings [?], where an adversary poisons the data used for clustering while seeking to maximize an objective function:

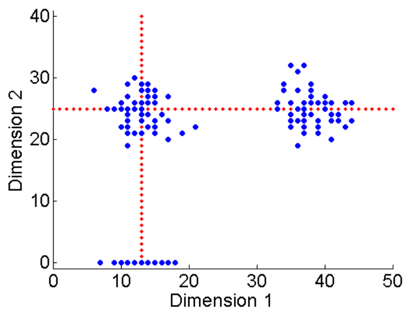
$$g(\mathcal{A}') = d_{\mathcal{C}}(\mathcal{C}, f_{\mathcal{D}}(\mathcal{D} \cup \mathcal{A}')). \tag{1}$$

For this objective function g, $\mathcal{A}'$ is a set of attack samples. In our case, these are activity sessions controlled by the attacker that are not necessarily aimed at compromising the web server, but rather at disrupting the unsupervised machine learning process. $\mathcal{D}$ is the uncontaminated dataset, and $\mathcal{D} \cup \mathcal{A}'$ represents the poisoned dataset, $\mathcal{D}'$. $f_{\mathcal{D}}$ represents the output of clustering $\mathcal{D}'$ but with a restriction to the samples in $\mathcal{D}$. $\mathcal{C}$ represents an uncontaminated clustering of $\mathcal{D}$, and $d_{\mathcal{C}}$ represents a distance measure between the uncontaminated clustering and the clustering obtained after adversarial action. This maximization problem i non-trivial to solve in general situations, and prior work has focused on developing hueristics for solving this problem for single linkage and complete linkage agglomerative clustering [?, ?, ?]. Because these heuristics do not apply to DBSCAN, we evaluate these attacks through simulations. We utilize the 8-feature set for our adversarial simulations due to the simplified geometry and more manageable cluster count of this feature space.

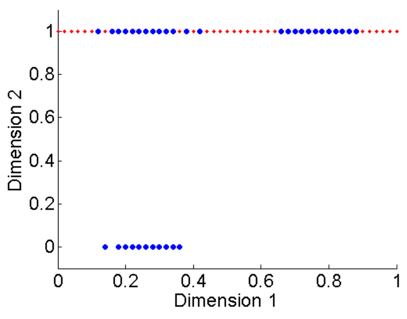
Cluster Poisoning Attacks. In a *bridge attack* [?,?,?], the adversary seeks to merge all true data samples into a single cluster with as few attack samples as possible. The benefit of this attack, even if a single cluster state is not reached, is that a reduced cluster count that entails more broadly defined clusters potentially allows for the attacker's targeted malicious actions to be misidentified as normal activity. We extend this attack strategy to DBSCAN. This involves injecting sessions that are less than ϵ apart from each other, in order to form bridges between all clusters. We further propose a novel poisoning attack, where the adversary seeks to cover a large volume of the feature space with a *net* (a grid in the feature space) of points less than ϵ apart. We call this the *net attack*. In an extreme scenario, where all the features are numerical and the minimum ϵ value under consideration is larger than the minimum increment of each feature values, the

		Clusters						
Category	Noise	1	2	3	4	5	6	7
Status 403	118	798	39	18	20	0	224	10
Status 404	86	7237	46	22	180	0	2	8
Admin	29	0	1407	0	183	0	0	0
Login	58	0	0	0	183	0	0	0
DB Add	10	0	0	0	0	0	228	0
Misconfiguration	94	252	0	18	0	0	0	10
robots.txt	40	6059	0	42	17	7	2	41
Total	235	144484	1407	121	183	49	228	144

Table 4: DBSCAN Cluster Counts for Reduced Feature Set.



(a) Numerical features.



(b) Binary and numerical.

Figure 2: Identical simulated data with different feature representations showing the true data in blue and the adversary's injected attack data in red.

adversary can completely compromise DBSCAN clustering by stationing at least one attack point at every discrete location in the feature space. In such a completely connected feature space, DBSCAN outputs only one cluster and no noise samples. We developed two variants of this attack for our simulations. In a *narrow net* attack, the adversary injects data points in the range of the existing feature values (this is also the case for the bridge attack). In a *wide net* attack, the adversary randomly injects points that lie outside these ranges, which rescales the feature space and causes DBSCAN to cluster together points that were previously considered far apart. This suggests that the choice of feature representation has a significant impact on the adversary's success.

Cluster Poisoning Defenses. The attacks in $\mathcal{A}'$ are subject to *constraints based on the feature space*, such as upper and lower bounds and limits related to discretization of the feature space. The intuition behind our proposed defense is to design a feature space that takes advantage of these constraints to prevent attacks against DBSCAN clustering. Without proper feature space design and selection of neighborhood distance ϵ, bridging and net attacks can have a severe impact on DBSCAN clustering, particularly in the scenario which we consider here where the adversary can potentially use a large number of distinct IP addresses to add a large number of attack samples to the dataset.

To defend against net attacks, we exploit the poor scalability of these attacks and design a feature space that is too large for the adversary to tile with attack points. For example, while an adversary might launch 10^6 attacks to completely tile a 3 dimensional feature space where each feature has 100 discrete possible values, most adversaries would be unable to launch 10^{20} attacks in order to tile a 20 dimensional feature space with 100 values per dimensions. Nevertheless, knowledge or accurate assumptions about the underlying data distributions can improve the effectiveness of both net and bridge attacks even in high dimensional feature spaces.

A more effective defense is to construct a feature set with both *binary* and *numerical* variables to prevent cluster merging. Building on the observations from §3, we represent our keyword features—deemed most important by the human analysts—as binary features, and we represent the features given secondary consideration as numerical variables. We then restrict the neighborhood size ϵ such that the maximum value of ϵ is less than the binary value differences in our feature space. Consequently, if two activity sessions have different values for a binary variable in the feature space, then this restriction causes these two activity sessions to necessarily lie in different clusters regardless of adversarial manipulations. This places an upper bound on the adversary's objective function in Equation 1 and prevents the adversary from merging all sample points into a single cluster. Figure 2 illustrates this defense. In the left panel, both features are numerical, which allows the adversary to bridge the 3 clusters together for either agglomerative clustering or DBSCAN. In the right panel, feature 2 is converted from a numerical to a binary representation and feature 1 is scaled between 0 and 1. With this feature representation, the adversary cannot bridge data points that are separated by the binary levels of feature 2.

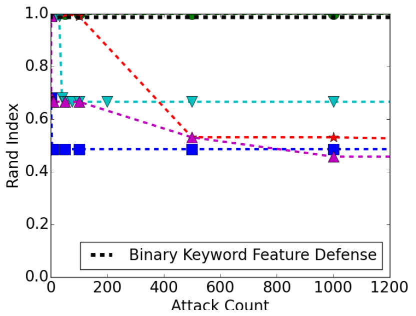
Evaluation of Attacks and Defenses. To assess the effectiveness of our attacks and defenses, we simulate the activity of several adversaries, with varying levels of knowledge, using a subset of 10,000 activity sessions.

We assume that the DBSCAN hyperparameters have been fixed prior to adversarial intervention; in our simulations, we set ϵ to 0.1 and the minimum sample number to 10. This parameterization allowed for a balance between cluster size and cluster number for our dataset. We use two variants of the 8 dimensional feature set described in §4.1: one variant uses fractional keyword features (scaled between 0–1), and the other uses binary features for "login," "admin," and database insertion keywords.

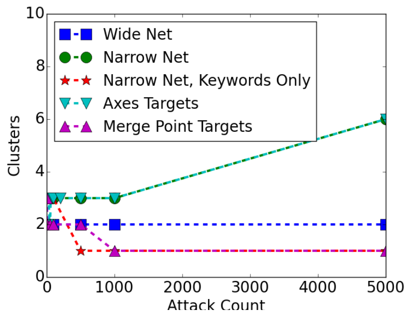
We assess attack impact by computing the adjusted Rand index between the initial clustering and the clustering for the original dataset after adversarial actions. Figure 3a illustrates the impact of 5 attacks based on the net and bridging attack strategies. We explore two variants of the net attack. For the narrow net attack, the adversary selects feature values are randomly selected between 0 and 10, for keyword, method, and status features, and selected between 0 and 3 for timing features. This narrow net attack is not effective (Rand index ≈ 1 for up to 5,000 injected sessions) because only a few out of the 10^7 points in the feature space successfully bridge the clusters. However, the narrow net attack can be restricted to injecting only 10^3 points related to keyword features, which results in a rapid degradation of the clustering quality. The second variant of the net attack seeks to rescale the feature space hypercube. In the original 10,000 sample dataset, the maximum keyword counts are below 20, and the selected ϵ cannot initially link clusters separated by more than 2 keyword counts (e.g. a session with 2 "admin" probes cannot be linked to a session with 0 "admin" probes). The wide net attack works against the numerical keyword features by adding sessions with large keyword counts, randomly selected in the interval between 0 and 100. Injecting these sessions into the dataset rescales the feature space unit hypercube, so that ϵ can link together two sample points even if they differ by a keyword count greater than 2. To defend against this rescaling we could fix the maximum value for each numerical feature, but this would also affect the normal activity clusters if the site grows in popularity.

Rapid clustering degradation is also possible if the adversary does not know the exact distribution of the dataset, but is able to guess some properties of these distributions. For example, in an axis-targeted attack variant, the adversary assumes that a cluster of normal activity exists where all features have zero values. Indeed, for our dataset, 93% of sessions have zero values for all features. To merge clusters, the adversary starts by tiling each axis of the feature space hypercube with attack samples. For example, the adversary can start by sending requests with 1–10 POST methods per session, followed by sessions with 1–10 403 status codes and then sessions with 1–10 404 status codes; in all these sessions, the features not varied are set to 0. Once each axis of the feature space is tiled in this manner, the attacker then adopts the strategy of the standard *narrow net* attack and randomly adds samples in the feature space in an attempt to merge clusters that do not lie along the axes. Similarly, for an attack variant that targets merge points, the attacker begins by attempting to merge the normal activity cluster with malicious clusters by injecting points with low keyword feature values and zero values for the other features. An appropriate point selection for our data can degrade the Rand index by 25% with only a single session injected into the dataset. Then, after these initial merge point guesses are made, the attacker defaults to the *narrow net* strategy for additional injection attacks related to keyword features.

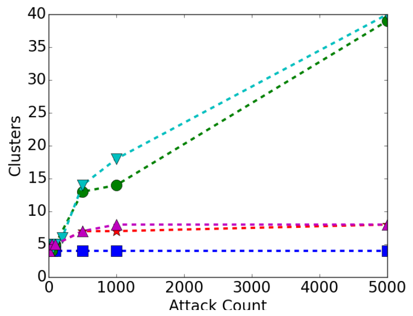
When using binary keyword features—which corresponds to our defense from §4.3—the Rand index does not drop below 0.98 for any of these attacks; this is illustrated with the black dashed line in Figure 3a. However, this is not the only indicator of attack damage. Because the results of clustering are interpreted by human analysts, we must also consider the cluster and noise counts after attack sample injection. Figure 3b shows that the axis-targeted and standard *narrow net* attacks allow the adversary to linearly increase the cluster count for both our feature sets; however, the increase is steeper with binary keyword features, as shown in Figure 3c. Similarly, the *wide net*, axis-targeted and standard *narrow net* attacks cause a linear increase in the number of noise samples when utilizing numerical features, but only the *wide net* has this effect with binary keyword features. These additional clusters and noise samples correspond to attack sessions, as the high Rand index suggests that the original clusters are not perturbed significantly by the adversarial interference. In consequence, systems that rely on DBSCAN for attack detection must carefully consider the trade-offs among all the performance metrics as no feature representation strategy is able to prevent all the potentially undesirable effects that an adversary may induce.



(a) Impact of attacks on adjusted Rand index.



(b) Cluster counts with numerical features.



(c) Cluster counts with binary keyword features.

Figure 3: Effectiveness of attacks and defenses, for DBSCAN clustering.

5 Threat Models

The human analysis process described in Section 3.1 and the process of designing and tuning the clustering-based attack detector from Section 4.1 provide blueprints for adversaries aiming to bypass combined human and automated defenses. In particular, the sequential steps in the two processes provide opportunities for perturbing the final results. Building on our analysis of the effectiveness of human and automated attack detection, in this section we discuss the factors that may degrade this effectiveness and concrete strategies that the adversary may pursue.

5.1 Threat Model for Manual Attack Detection

To understand the threats against human-driven detection, we must consider the goals and capabilities of the adversary, as well as possible attacks against the human analysis process.

Attacker goals. An attacker may pursue two distinct goals. First, an attacker may try to bypass the human analyst by crafting attacks that are difficult to distinguish from normal activity. Second, an attacker may try to hide the real target of the attack, even if the analyst suspects that malicious activity is present in the logs, by distracting the analyst or by obfuscating the attack vector. Both attacks aim to *reduce* the effectiveness of the manual detection process.

Attacker capability. There are three main components of an attacker's potential knowledge:

- *Human resources available*: At a gross level, the attacker may infer some information regarding the expertise and amount of human resources devoted to security analysis. Whereas a financial institution or government defense organization will likely have significant expertise and resources, a small-business website or local government system may have fewer resources available. Thus, the attacker can potentially induce a high number of false alarms knowing that there will not be time to investigate them all, or use particularly obtuse attacks to confuse the novice sysadmin.

- *Web technologies used*: The technologies used to serve a website are typically apparent by examining URL structures, HTML meta tags, or even website footers. Similarly, whether a website allows user content (forums, comments, uploads) vs. serving mostly static content can be ascertained by browsing the site. Such information is useful in targeting known vulnerabilities in webserver technologies as well as understanding what ”typical” behavior on the website might look like in order to make malicious activity less apparent.

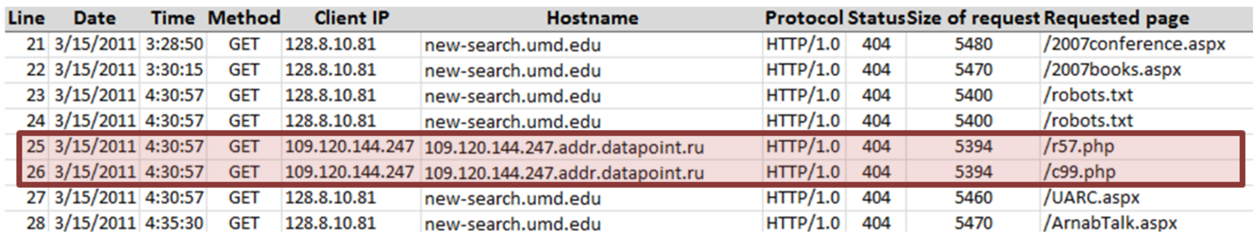
- *Traffic patterns*: The purpose and audience of a website is generally apparent from a site visit. Whereas a search engine may serve the entire world, a local credit union may only have customers from a specific region, and a small-business webmail server only expects requests from a few individuals. Further, inferring the geographical location of a website's clientele can be useful for identifying periods of high or low activity. Any understanding of the traffic patterns of a website benefits the attacker if they can make malicious requests resemble ”normal” activity, i.e., by restricting malicious activity to high activity periods and originating from an expected geographic area via a VPN service.

Each of these knowledge components in turn impacts the attacker's ability to launch attacks against the human and automated components of the target's defensive system. For attacks specifically targeting human analyst weaknesses, we assume that the attackers are aware of the process described in §3.2, but are not able to enumerate all the hypotheses that the analysts will formulate and test.

Attack strategies. Human analysts are vulnerable to both cognitive biases and information processing limitations that can be exploited by attackers. Human information processing can be characterized as *resource-limited* and *data-limited* [?]. In the case of log file processing, the quality of the data available (data-limited) and the amount of information to be critically examined (resource-limited) impact the analyst's ability to accurately identify malicious activity. Attacks on *human resource limits* target the *filtering* and *searching stages* of analysis. For example, embedding a few targeted attacks among a large number of innocuous requests increases the amount of data that must be accurately processed by the analyst. Obfuscating URI query strings with percent encoding may alert the analyst that the log entry contains malicious activity, but this also increases the processing required to understand the intent of the queries (second attacker goal). Attacks on *human data limits* purposely conceal information content so that the analyst cannot make an accurate decision and can affect *all three stages* of analysis. For example, IP spoofing [?], user agent manipulation [?], and using VPNs or proxies would all prevent accurate information from reaching a human analyst (first attacker goal). Beyond explicit information hiding techniques such as these, data limits affecting human analysis is often governed by the quality of log files and the analysis tools available to the human analyst.

The adversary may exploit another weakness in human analysts called *anchoring* – the tendency to draw conclusions that are biased toward the initial information observed regardless of the other information available [?]. This attack strategy involves injecting attacks that mask other attacks. For example, an attacker could take advantage of the status code feature (e.g., searching for 403 and 404 status codes) used by many analysts in our experiment. The excerpt in Figure 4 is from one of six, 50-line log file excerpts shown to users during the experiment and contains two malicious lines attempting to access known PHP backdoor shells. Note that these requests are embedded among other 404 response requests from a search crawler with an outdated cache. In the experiment, the users performed significantly worse in correctly identifying whether the log excerpt contained malicious activity or not than all other log files (55% correct in this file vs. an average of 79% for other files). Other participants in the log file study erroneous associated requests from bots and proxies with malicious intent regardless of the overwhelming evidence that these requests were from benign search crawlers [?]. Attacks on the anchoring bias lower the utility of the *filtering* and *searching* stages.

In *alarm fatigue* [?, ?] humans become desensitized to warnings. An adversary can induce alarm fatigue by launching harmless attacks and scans that cause false alarms (a primary concern for intrusion detection) or by injecting benign traffic in order to reduce the rate of true positive alarms. An analyst searching for particular feature, such as the admin/login keyword, will rapidly become desensitized to such attempts if they are identical and unsuccessful (e.g., from an outdated exploit kit). Following tens or hundreds of such



Line	Date	Time	Method	Client IP	Hostname	Protocol	Status	Size of request	Requested page
21	3/15/2011	3:28:50	GET	128.8.10.81	new-search.umd.edu	HTTP/1.0	404	5480	/2007conference.aspx
22	3/15/2011	3:30:15	GET	128.8.10.81	new-search.umd.edu	HTTP/1.0	404	5470	/2007books.aspx
23	3/15/2011	4:30:57	GET	128.8.10.81	new-search.umd.edu	HTTP/1.0	404	5400	/robots.txt
24	3/15/2011	4:30:57	GET	128.8.10.81	new-search.umd.edu	HTTP/1.0	404	5400	/robots.txt
25	3/15/2011	4:30:57	GET	109.120.144.247	109.120.144.247.addr.datapoint.ru	HTTP/1.0	404	5394	/r57.php
26	3/15/2011	4:30:57	GET	109.120.144.247	109.120.144.247.addr.datapoint.ru	HTTP/1.0	404	5394	/c99.php
27	3/15/2011	4:30:57	GET	128.8.10.81	new-search.umd.edu	HTTP/1.0	404	5460	/UARC.aspx
28	3/15/2011	4:35:30	GET	128.8.10.81	new-search.umd.edu	HTTP/1.0	404	5470	/ArnabTalk.aspx

Figure 4: Excerpt from log file showing the r57 and c99 shell access requests.

false alarms, the single, successful admin exploit attempt targeting a zero-day vulnerability could easily go unnoticed. This attack strategy primarily affects the *search* stage.

Human analysts are also susceptible to the *clustering illusion* where a series of events may seem significant when viewed within a small time frame but are actually random against the whole population of events [**?**, **?**]. In the log file experiment, some subjects were suspicious of frequent and repeated requests from the same client IP or hostname, regardless of whether these requests were malicious or not. The implication for an attacker is to strive for the converse: whereas a burst of malicious behavior would be visually apparent in a log file, long-period malicious queries interspersed benign queries from the same source would be difficult for a human to discern.

We note that effective and ineffective analysts employ different processes and features. The poor performers were distracted by requests originating from automated agents, such as Web crawlers, incorrectly marking them as malicious. In contrast, experienced admins were more suspicious of human actors. Novice analysts also put undue weight on the timing of requests and often marked a series of rapid requests as potentially malicious. This suggests that knowledge of the analyst expertise level can provide an advantage to the attacker. For example, to defeat a novice analyst, the attacker could space out the requests or exploit the clustering illusion by generating distracting bursts of activity.

5.2 Threat Model for Automated Attack Detection

Clustering algorithms like DBSCAN are sensitive to the choice of features and hyperparameters, as illustrated in §4.1. An attacker can exploit this weakness to prevent the algorithm from flagging new, previously unknown, attacks.

Attacker goals. An attacker may pursue two distinct goals. The attacker's *primary goal* is to compel the unsupervised learning system to include the attack sessions into a cluster that corresponds to normal activity. In this case, the automated technique provides *no benefits* for detecting new attacks. The attacker's *secondary goal* is to increase the number of clusters and or the noise counts, to overwhelm the analysts who must inspect the clusters and the anomalies to determine if they correspond to new attacks. In this case, the attack may still discovered, but the likelihood that this will happen decreases. In consequence, if the attacker achieves the secondary goal, the automated technique provides *reduced benefits* for detecting new attacks.

Attacker capability. The attacker's knowledge has three components:

- *Features*: Knowledge of the features selected for clustering points the attacker to the request fields that must be manipulated in an attack. In general, a stronger attacker will possess a more complete knowledge of the feature set utilized. In general, it is reasonable to assume that an attacker can make educated guesses about some of the features (e.g. 403 and 404 status codes), based on common analysis practices.

- *Feature distributions*: Knowledge of the feature distributions allows an adversary to craft attacks that resemble normal behavior. An attacker may not possess a perfect knowledge of all the feature distributions; for example, the geographical distribution of requests for the Drupal site we consider in our experiments depends on Fraunhofer's current clients, which are confidential. However, the attacker may guess the approximate ranges of numerical features, or the fact that benign sessions typically have low numerical values for most features.

- *Algorithm and hyperparameters*: Knowledge of the algorithm can significantly influence the adversary's attack strategies. For example, a strategy that would be effective against anomaly detection, such as generating a large number of false positive outliers, would be less effective against a clustering algorithm capable of grouping these false positives in a single cluster. Additionally, a bridge poisoning attack strategy specifically targeted against single linkage agglomerative clustering [?] would potentially be less effective against Gaussian mixture model clustering. Knowledge of the feature space geometry, which results from the feature distribution and representation and the choice of algorithm hyperparameters, allows an adversary to conduct *poisoning* attacks [?], which alter the cluster composition and location.

The adversary may utilize a large number of distinct IP addresses (e.g., from a botnet) to inject attack samples to the dataset, but does not have the ability to alter or delete the recorded log sessions that were initiated by other users. The corruption of log sessions recorded prior to a breach can be prevented using secure logging techniques [?]. We further assume that the clustering hyperparameters are selected based on domain knowledge or based on a training data set that the adversary cannot poison. In our simulations from §4.3, we evaluate the effectiveness of adversarial strategies with varying levels of knowledge about the feature space and the underlying data distributions.

The features used for clustering have varying degrees of resiliency to adversarial manipulation. For example, the user agent string in web logs is completely under the control of the client and can typically be arbitrarily manipulated without impacting the effectiveness of potential attacks. Similarly, timing pattern features can be correlates of attacks, but they lack a causal relationship except for denial of service cases. As a result, an adversary can arbitrarily manipulate these patterns in order to avoid detection. This can rate limit the attacker, but it does not deter attacks in general. In contrast, many keyword features are causal and, therefore, more robust to adversarial manipulation. If an adversary seeks to access a login or administration page, then it is typically necessary to include these keywords in the request or at the very least a percent encoded alternative. Hence, while the attacker cannot construct an exploit against the server with arbitrary features, we assume that the adversary can construct a session with arbitrary features for disrupting the clustering algorithm.

Attack strategies. The adversary may conduct the bridge and net attacks described in Section 4.3. The bridge attack exploits knowledge of the features and the feature distributions to inject attack samples that effectively link clusters together in the feature space. For DBSCAN, this involves adding attack samples less than ϵ apart from each other in order to form bridges between all clusters. In consequence, the attacker must also know the approximate values of the clustering hyperparameters to conduct an effective bridge attack against DBSCAN. This attack is illustrated for simulated data in Figure 2.

The *net attack* is practical when the adversary has knowledge of the the feature space, or at least a subset of the feature space, but limited knowledge of the value distributions in the feature space. In a *narrow net* attack, the adversary knows the range of the feature values and tries to inject merge points that degrade the existing clusters. To create a complete net, the adversary must inject $(R/\epsilon)^n$ points, where n is the number of features and R is the range of feature values, which may be prohibitively expensive. The number of points injected in the feature space may be pruned by exploiting partial knowledge of the feature distributions; at the extreme, when the adversary has perfect knowledge of the distributions, this strategy is the same as the bridge attack. However, the adversary may also conduct a *wide net* attack, which only requires knowledge of the feature ranges. In this attack, the adversary randomly injects points that lie outside these ranges, which rescales the feature space and causes DBSCAN to cluster together points that were previously considered far apart.

The effectiveness of these strategies can be further improved if the adversary knows specific properties of the feature distributions. For example, if features with zero values are common in the data set, the adversary can inject sessions that vary one feature and set the other features to 0, to merge clusters that lie along the axes of the feature space. The evaluation results suggest that our proposed defense can prevent cluster merging, thwarting the adversary's primary goal. However, the defense may facilitate the adversary's secondary goal by producing a large number of clusters and noise data points that must be interpreted by human analysts.

Additionally, the outputs of machine learning systems are often difficult to interpret [?]. For example, DBSCAN outputs a set of clusters and noise data samples, but does not outline the reasoning behind these inferences. In consequence, there is a *semantic gap* between the outputs of machine learning techniques and their operational interpretation [?]. Security analysts may interpret the clustering results by analyzing the core points in each cluster (as we did in our experiment), but this task is subject to the attacks against manual analysis discussed in Section 5.1.

6 Guidelines and Open Questions

Building on the insights from our research, in this section we provide guidelines for combining human analysis and automation effectively. We also identify areas where further research is needed.

Humans excel at recognizing patterns, and abnormalities in these patterns, which gives them the ability to infer the meaning of unusual and previously-unseen behavior. However, a savvy adversary can attack human analysis processes. Automated systems can potentially help human analysts overcome these weaknesses:

- *Human resource limitations*: The best defense against resource limitations is to minimize the number of suspicious activities that humans must investigate. This can be difficult, especially if the adversary intentionally seeks to overwhelm the analysts. Another defense is to provide the analysts with threat intelligence tools that reduce the manual effort that humans must expend to understand the presented data. For example, percent decoders for URLs and IP lookup services would have helped the analysts in our user study.
- *Human data limitations*: Improving the quality and completeness of the recorded data (e.g., ensuring that useful fields are logged) will improve the performance of human analysts. This requires understanding what fields the analysts find useful. Additionally, systems that automatically synthesize data from disparate sources (e.g., log data and network statistics) unburden this responsibility from the analyst.
- *Anchoring*: Increased training for analysts to set aside past biases and examine new behaviors is a straightforward strategy to overcome this issue.
- *Clustering illusion*: Automated systems can keep a history of traffic, and a system that helps an analyst compare a selected block of unusual behavior against normal traffic patterns across a website's history can help dispel the clustering illusion.
- *Alarm fatigue*: An attacker intentionally generating false alarms could be defeated by an automated system that accepts analyst feedback to selectively ignore features for a period of time or to cluster the spurious alarms with other likely attack sessions.

Clustering the data with an unsupervised learning algorithm, such as DBSCAN, can be an effective way to focus the analyst's attention on the data points that are representative of larger classes of activities or that are outliers. As these algorithms are vulnerable to cluster poisoning attacks, a defensive mechanism (such as the one proposed in Section 5.2) should be deployed. A more subtle problem is that, because of the semantic gap between the clustering results and their interpretation, it is difficult for security analysts to decide when to trust the outputs of the machine learning model. For DBSCAN, this problem could be addressed by showing the analyst a few core data points from each cluster; our experience suggests that this makes it easy to identify the activity represented by the cluster. Finally, even if a clustering algorithm such as DBSCAN is tuned to detect previously unknown attacks, such attacks must still occur first in order for the automated system to detect and identify these attack variants. In contrast, human analysts who inspect the raw data can utilize their intuition to identify novel attacks and mitigate them before they are observed in the wild, as illustrated by our vulnerability discovery presented in Section 3.3.

Ultimately, understanding the effectiveness of hybrid detection systems, which aim to provide the best of both the human and machine worlds, requires considering consider both the human and the machine components. Such an active framework should specify the interactions between the clustering system and the human analysts, e.g. to filter out noise samples that do not correspond to attacks. Prior research in adversarial unsupervised learning focuses only on adversaries aiming to merge clusters, by modeling the adversary's objective function in terms of the distance between the original and the poisoned clusters. Therefore, a challenge for future research is to model the human component of the system and to define an objective function that reflects the success of the hybrid detection system. Another research direction concerns the ability to detect the injected poison samples, in order to remove them in a filtration or pre-processing step prior to clustering. For example, a historical comparison of the number and distance between and within clusters may reveal that the number of clusters, or the average distance between them, has changed. It may also be possible to adopt an iterative approach by further clustering within the larger clusters. More generally, the poison samples may differ from the statistical distribution of normal user activity. Such data filtering could limit the capacity of an adversary to conduct attacks involving new behaviors. In general, if the defenders know the algorithm used by the attacker to craft poison samples, then the attack samples could become easy to detect. However, the adversary can modify the algorithm slightly in order to evade this detection, leading to an arms race. Consideration of any such filtering or pre-processing protocol would involve escalated gamesmanship between the security analysts and their adversaries as the adversaries would have to continuously adjust and adapt their attack strategies to more subtly deviate from normal user expectations. These approaches also need not be limited to unsupervised machine learning or DBSCAN, as we have presented here, and future hybrid human-machine learning approaches might have potential benefits when incorporated into semi-supervised and supervised machine learning analysis pipelines as well.

7 Related Work

There are parallels between our work and the software engineering concept of program comprehension [?], which observes that developers look for beacons in the code that appear relevant to the programming task, similar to IT security administrators searching for patterns indicative of anomalous behavior in a log file [?]. We describe an abstract process that describes the work of security analysts and a threat model derived from this process (§3).

Unsupervised learning techniques have been utilized extensively in security, for example to group malware samples into malware families [?,?,?,?]. Several researchers have applied clustering to log data, and have used some of the features we consider in our work, for detecting attacks and misconfigurations. Chapple et al. [?] applied expectation maximization (EM) clustering, with a geographical distance metric, to authentication logs in order to identify violations of institutional policies. Zhang et al. [?] extended this approach with additional features to identify compromised accounts. Yen et al. [?] applied an adaptation of K-means clustering to a variety of enterprise logs (e.g. authentication, DHCP, outgoing Web requests) to identify malware infections and policy violations in a large enterprise. In this paper, we contrast the strategies and features employed by human analysts and clustering systems for detecting attacks in order to characterize the strengths and limitations of each approach.

Sommer et al. [?] discussed the difficulties of applying supervised machine learning techniques to network intrusion detection and anomaly detection problems. Kloft et al. [?] explored anomaly detection under adversarial interference. Unsupervised machine learning in adversarial settings has been explored in [?, ?, ?, ?, ?], emphasis on agglomerative clustering and clustering based on singular value decomposition. This work focuses on disrupting the machine learning processes and does not consider the human component of analysis. We extend this body of work with new attacks and defenses for DBSCAN clustering, and we reason about the impact of human intervention on adversarial strategies.

8 Conclusion

We describe the process by which human analysts discriminate between malicious and benign activity in Web server logs and the features they utilize in this process. We discuss strategies for bypassing the analysts, when an attacker wants to remain undetected or to hide the real target of the attack. We show that some features that are useful for human analysts are not useful for DBSCAN clustering, because they result in a poor feature space geometry, have a high potential for false positives or are indirect indicators for attacks. We propose a threat model for DBSCAN clustering with adversarial interference and we describe two attacks, which aim to merge benign and malicious clusters or to cause an increase in the cluster and noise counts. We also propose a defense, which relies on choosing a feature representation that constrains the adversary's ability to manipulate the feature space. Our evaluation suggests that our proposed defense prevents an adversary from creating clusters that are highly dissimilar from the clusters produced in the absence of adversarial interference, and provides some protection against inducing high noise counts, but it may also help the adversary inflate the cluster counts. Finally, we discuss the implications of these insights for integrating unsupervised learning and manual analysts, and we highlight promising research directions.

Acknowledgement

Carl Sabottke, Daniel Chen, and Tudor Dumitraș were partially supported by the National Science Foundation award CNS-1564143 and the Maryland Procurement Office contract H98230-14-C-0127. Carl Sabottke received graduate student support from National Science Foundation Graduate Research Fellowship 2014161339. Lucas Layman was supported by an internal grant from Fraunhofer USA.