Detecting Malicious Web Scraping Activity: a Study with Diverse Detectors

Pedro Marques
LaSIGE, Faculdade de Ciências
Universidade de Lisboa, Portugal
and
Centre for Software Reliability
City, University of London, UK
pedro.daniel@city.ac.uk

Zayani Dabbabi
Amadeus, France
zayani.dabbabi@amadeus.com

Miruna-Mihaela Mironescu
Amadeus, France
miruna-mihaela.mironescu@amadeus.com

Olivier Thonnard
Amadeus, France
olivier.thonnard@amadeus.com

Alysson Bessani
LaSIGE, Faculdade de Ciências
Universidade de Lisboa, Portugal
anbessani@ciencias.ulisboa.pt

Frances Buontempo
Centre for Software Reliability
City, University of London, UK
frances.buontempo@city.ac.uk

Ilir Gashi
Centre for Software Reliability
City, University of London, UK
ilir.gashi.1@city.ac.uk

***Abstract* — We present results on the use of diverse monitoring tools for the detection of malicious web scraping activity. We have carried out an analysis of a real dataset of Apache HTTP Access logs for an e-commerce application provided by a large multinational IT provider for the global travel and tourism industry. Two tools have been used to detect scraping activities based on the HTTP requests: a commercial tool, and an in-house tool called Arcane. We show the benefits that can be achieved through the use of both systems, in terms of overall *sensitivity* and *specificity*, and we discuss the potential sources of diversity between the tool's alert patterns.**

Keywords— design diversity, malicious web scraping, botnet detection, security assessment

I. INTRODUCTION

Web scraping is the process of using bots to extract content and data from a website[1]. There are many legitimate use cases of web scraping, such as a search engine bot crawling a site, analysing its content and then ranking it; price comparison sites deploying bots to auto-fetch prices and product information for seller websites, etc. However, web scraping is also used for illegal purposes. Use cases of illegal malicious web scraping include undercutting of prices, theft of copyrighted content, etc. In price scraping, a perpetrator typically uses a botnet from which to launch scraper bots to inspect competing business databases. The goal is to access pricing information, undercut rivals and boost sales. Attacks frequently occur in industries where products are easily comparable, and price plays a major role in purchasing decisions. Victims of price scraping can include travel agencies, ticket sellers and online electronics vendors. Large multi-nationals, in particular those active in specific sectors (such as e-commerce, gambling, travel, etc), are prime targets for this type of malicious activity.

One way of detecting malicious web scraping activity is to look for evidence of the use of botnets. Resources needed to run web scraper bots are substantial, due to the large amount of data collected from any one site. Attackers, who lack such resources, often resort to the use of botnets to carry out these attacks. Finding these botnets can be done by looking at the commonalities observed between different connecting clients, because these different clients will be infected with the same malware. The operation mode of the malware will yield similar requests from different clients, thus pointing to the existence of botnets.

To protect themselves from malicious web scraping attacks, organizations use specialized software that can monitor for suspicious activity, attempt to separate bot traffic from human traffic, use IP reputation websites to block activities from suspicious IP addresses, monitor the behaviour of visitors in the way in which they interact with the website to check for abnormal browsing patterns.

In this paper we provide results from an analysis of a large dataset in which we analyse a commercial tool (we have anonymized the name and will refer to it as CommTool for the rest of the paper) and an in-house tool called Arcane developed by a large multi-national in the global travel industry. Both of these tools monitor the same application layer interactions to detect malicious web scraping behaviour. The industrial partner that provided the dataset carefully labelled the traffic into malicious and benign, hence enabling an analysis with the conventional metrics of analysing the performance of binary classifiers, namely sensitivity and specificity[2].

Our primary focus in this analysis was to investigate how diverse these tools are in their behaviour. We find there is significant diversity in the detection results of the tools. We then investigated further the causes of the diversity in the results (the design and configuration of the tools) and provide a summary of these findings. The results will be of use to practitioners in various industries that need to deal with advanced malicious web scraping and botnet activities, and how the use of diverse tools may help them counteract these threats. To the best of our knowledge, a similar study on the use of diverse tools for malicious web scraping has not been published before.

The rest of the paper is organized as follows: Section II presents related work. In Section III we provide a description of the dataset we have analysed, as well as the monitoring tools present in that dataset. In Section IV we start by giving an overview of our initial results. We then describe the diversity we have found between both tools employed in the dataset, and finally we attempt to provide an explanation for

[1] https://www.incapsula.com/web-application-security/web-scraping-attack.html

[2] https://en.wikipedia.org/wiki/Sensitivity_and_specificity

the diversity we observe in the alerting patterns of both tools. Finally, in Section V we discuss the implication of our findings, as well as our recommendations based on the results we observed and conclude with future work related to our analysis and findings.

II. Related Work

The security community is well aware of diversity as potentially valuable [4, 5]. Discussion papers argue the general desirability of diversity among network elements, like communication media, network protocols, operating systems, etc., but only sparse research exists on how to choose diverse defences (some examples in [5-8]).

There have been several works that have looked at ways in which malicious web scraping activities can be detected (with the use of data-mining algorithms [1], probabilistic and machine-learning models [2], as well as more general algorithmic approaches [3]), but none that we are aware of that has looked at combining multiple diverse detectors.

Potential benefits from design diversity for safety and reliability have been studied for many years. See for example work on a probabilistic model of diversity outlined in [13], [10] which were motivated by the work on N-version programming [14]. It has been discussed as a risk reduction strategy, particular at the start of a project [9]. The authors of [9] also warn that different application areas require different measures to calculate the effectiveness of design diversity, and as such the publishing of results from one area might not be directly applicable in other areas. Littlewood et al. further compound this point by discussing how measuring the performance of IDSs should be done based on categories of attacks, rather than using an average mixture of attack classes [4].

Work has also been done on providing statistical measures of the diversity of ensemble methods, particularly in the case of binary classifiers [11, 12].

III. Description of the Dataset and Tools

The dataset consists of Apache HTTP Access logs for an e-commerce application. The application is a fairly typical electronic retail application in the travel industry. The dataset covers a period of 5 days, from May 7th to May 12th, 2018. Two tools, CommTool and Arcane have been used to detect scraping activities based on the HTTP requests.

A. CommTool

The company that provided the dataset uses a version of CommTool deployed in the cloud in front of the web servers of the web application to protect. This means that all HTTP requests coming from users are first inspected by CommTool. Legitimate requests are forwarded to the web application and requests deemed from bots are blocked.

CommTool (as well as many other bot detectors) use different techniques to detect scrapers:

- **Client-side fingerprinting** – A JavaScript file is downloaded from the protected website and run on the client's browser. This script extracts many device attributes to create an accurate fingerprint of the client's system. The fingerprints are shared worldwide amongst CommTool products, creating a global database of known violator fingerprints. Bots and scrapers are detected based on session attributes, such as session length, pages per session, pages per minute, etc.
- **JavaScript tests** – For suspicious user sessions, CommTool can run further client-side JavaScript tests, such as inspecting the consistency of device attributes.
- **Machine learning** – CommTool uses evolving behavioural user models based on the data collected from different domains protected.
- **Custom rules** – For advanced scraping activities that are not detected using the above methods, CommTool can implement custom rules based on user device attributes. Custom rules can also be created on request by the customers of the tool (for example to monitor a particular domain of interest to the customer).
- **Known violator databases** – CommTool uses a worldwide database of known violators for easy identification of bots. The known violators can be IP addresses, subnets, ISPs and countries.

B. Arcane

Arcane is an internal tool to the company that provided the data, used to detect scraping activities. Arcane is used to monitor CommTool's performance for domains which are already protected by CommTool and assess robotic activities for non-protected domains. Arcane uses only Apache HTTP access logs and the information these contain to detect scraping activities. HTTP access records are grouped into HTTP sessions via a unique session identifier. The unique session identifier is stored in the client browser cookies and logged in the Apache audit trails. The features are collected from these sessions are used to detect bot activity. The session features used are shown in the table below.

TABLE I. Arcane session features.

Feature	Description
Session Duration	The time elapsed between the first and last HTTP request in a session
Number of requests per session	The number of dynamic requests during an HTTP session
Burst rate	The maximum number of requests per time unit, made during a sliding time window
Static/Dynamic request ratio	The ratio between the number of requests to static resources and the number of requests to dynamic pages
Entropy	A measure of the diversity of requests paths in session
Number of IP addresses	The number of distinct IP addresses used during the same session
Number of user agents	The number of distinct user agent strings in a session
Number of HTTP errors	The number of requests that results in 4xx or 5xx error pages for a session
Ratio availability request	The ratio between the number of requests made to sensitive pages and the total number of dynamic requests during a session

For each of the features in Table I, a manually defined scoring function is used to compute an anomaly score. These scores are then aggregated into a global anomaly score used to decide whether an HTTP session is legitimate or malicious.

The company that provided the dataset uses a custom known violator database, similarly to CommTool. While the dataset provider uses CommTool in blocking mode, every HTTP request is still observed by both tools, with Arcane analysing the traffic after the fact.

IV. RESULTS

A. Overview reuslts

Table II shows the total number of requests present in the dataset, as well as what portion of these were web scraping attacks. Table III shows how both Arcane and CommTool classified HTTP requests that were deemed as web scraping attacks (4,825,011). Numbers under "reported" identify true positives given by each tool, while "not reported" identify false negatives. The same information is shown in Table IV, for HTTP requests that were deemed as non-scraping traffic, where "reported" identifies false positives, and "not reported" identifies true negatives.

TABLE II. HTTP REQUEST COUNTS.

Total requests	5,028,429	(100%)
Total scraping requests	4,825,011	96%
Total non-scraping requests	203,418	4%

TABLE III. WEB SCRAPING REQUESTS ANALYSED BY BOTH TOOLS.

Web scraping requests		**Arcane**		**Totals**
		Reported	**Not reported**	
CommTool	**Reported**	1,369,281	146,998	1,516,279
	Not reported	1,430,200	1,878,532	3,308,732
Totals		2,799,481	2,025,530	4,825,011

TABLE IV. NON-SCRAPING REQUESTS ANALYSED BY BOTH TOOLS.

Non-scraping requests		**Arcane**		**Totals**
		Reported	Not reported	
CommTool	Reported	353	270	623
	Not reported	43,356	159,439	202,795
Totals		43,709	159,709	203,418

These numbers allow us to calculate sensitivity (positives correctly classified as such) and specificity (negatives correctly classified as such) rates for both tools, as shown in Table V. We see that Arcane is a more sensitive, but less specific system, meaning that it generates a higher number of both true and false positives.

TABLE V. ARCANE AND COMMTOOL SENSITIVITY AND SPECIFICITY.

	Arcane	**CommTool**
Sensitivity	58.02%	31.42%
Specificity	78.51%	99.69%

B. Diversity analysis

In monitoring environments adjudication systems are often used to decide whether observed items are malicious or not based on the decisions of multiple monitors. These are often described as "*r-out-of-n*" (*rooN*) systems, where an *r* number of systems, out of a total of *n* systems, need to raise an alert for it to be raised as an alert by the system. For our case study, where n=2, we can define *1oo2* and *2oo2* adjudication systems, which require just one or both systems (Arcane and CommTool) to raise an alert, respectively.

Based on the counts present in Tables III and IV we can extrapolate the results to 1oo2 and 2oo2 adjudication systems, in order to understand what kind of improvements can be achieved from using both systems in conjunction. We show in Tables VI and VII how both of these adjudication systems would have analysed web scraping and non-scraping HTTP requests, respectively. We extend this in table VIII, where we show the sensitivity and specificity rates for both adjudication systems. The changes in sensitivity and specificity when comparing each tool individually to the adjudication systems can be seen in Fig. 1 (the y-axis shows the absolute difference. e.g. 3.05 improvement in sensitivity observed by users of Arcane from switching to a 1oo2 system, is 61.07 – 58.02, etc.).

TABLE VI. WEB SCRAPING REQUESTS ANALYZED BY ADJUDICATION SYSTEMS.

	Reported	**Not reported**
1oo2	2,946,479	1,878,532
2oo2	1,369,281	3,455,730

TABLE VII. NON-SCRAPING REQUESTS ANALYZED BY ADJUDICATION SYSTEMS.

	Reported	**Not reported**
1oo2	43,979	159,439
2oo2	353	203,065

TABLE VIII. ADJUDICATION SYSTEMS SENSITIVITY AND SPECIFICITY.

	1oo2	**2oo2**
Sensitivity	61.07%	28.38%
Specificity	78.37%	99.83%

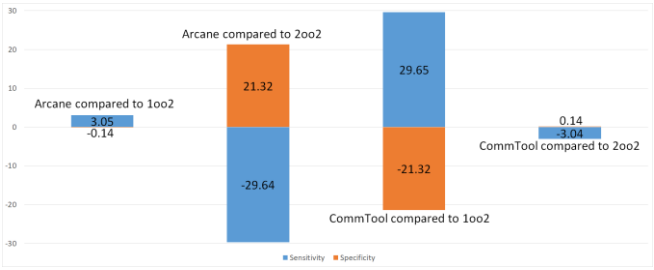

Fig. 1. Sensitivity and specificity changes from single tools to adjudication systems.

As expected, there is an increase in sensitivity in a 1oo2 system, in relation to both systems individually, while specificity shows a slight decrease when compared to Arcane, but a large decrease when compared to CommTool (roughly 20%). For a 2oo2 adjudication system we see the opposite, where sensitivity drops below that of either system individually, but specificity significantly increases. This is to be expected as:

- 1oo2 systems will, in *all* cases, perform:
  - *better or equal* to the best single system in the pair for *malicious traffic*, as any alarm from any of the two systems will lead to an alarm in a 1oo2 system;
  - *equal or worse* than the worst single system in the pair for *benign traffic*, as any alarm from either system for benign traffic will be incorrectly labelled as malicious.
- 2oo2 systems will, in *all* cases, perform:
  - *better or equal* to the best single system for *benign traffic* as the 2oo2 system only raises an alarm for benign traffic if both systems in the pair raise an alarm;
  - *equal or worse* than the worst single system in the pair for *malicious traffic*, as the 2oo2 system will only label an attack as malicious if both the systems in the pair label it as such.

What is important is *how much* better, or *how much worse*, would a diverse pair perform in these setups, and the results in Table VIII and Figure 1 give us some indications about this.

Based on the results presented here, it is clear that the tools complement each other in their detection capability, though which adjudication scheme to use would depend on the losses associated with the different types of failures (false positives and false negatives). From the perspective of a user running only one of these tools the results suggest that: users running Arcane should pair it with CommTool in a 1oo2 configuration, as they will see an improvement in sensitivity, with seemingly only a negligible deterioration in specificity. Users running CommTool can improve their sensitivity significantly with a 1oo2 adjudication system that runs Arcane alongside it, but this will lead to a large increase in false positives and hence an increase in cost (through additional personnel cost, for example).

C. Where does the diversity between the tools come from?

So far, we have demonstrated the presence of diversity in the alerting patterns of both tools but have not shown where this diversity is coming from, or why it exists. The way in which both Arcane and CommTool detect scraping activity, and the different techniques they employ is the main reason for this difference.

In Tables IX and X we show the number of CommTool and CommTool-only true positives based on the reasons CommTool used to flag the connections. In some cases, multiple reasons were used by CommTool to flag a request. In these cases, we have counted the alert in more than one category.

TABLE IX. COMMTOOL REASONS FOR TRUE POSITIVES (SHORTENED).

JavaScript Check Failed	1,422,528
JavaScript Not Loaded	797,420
JavaScript Not Loaded & JavaScript Check Failed	728,785
Rate Limited	671,081
Known Violator Data Center	157,920
Pages Per Session Exceeded	36,531
Session Length Exceeded	19,664

TABLE X. COMMTOOL-ONLY REASONS FOR TRUE POSITIVES (SHORTENED).

JavaScript Check Failed	91,408
JavaScript Not Loaded	90,740
JavaScript Not Loaded & JavaScript Check Failed	90,090
Rate Limited	77,567
Known Violator Data Center	26,750
Known Violator	5,866
IP Pinning Failure	1,299
Referrer Block	556

As explained before, CommTool is a primarily client-side tool that works by creating a fingerprint of the client's device and matching this fingerprint to a known violator database. This fingerprint is generated early on in a session, through the use of JavaScript. JavaScript checks are the most common reason for CommTool's true positives, including true positives only generated by CommTool. Because Arcane does not utilize JavaScript, this is theoretically a significant source of diversity between the tools. Nevertheless, Arcane is still able to detect most of the same connections detected by CommTool's JavaScript process (roughly 93.5%) since these connections are initiated by hosts that are in Arcane's blacklist.

The reasons for CommTool alerts are roughly the same between all CommTool true positives and CommTool-only true positives, with the exception of "Pages Per Session Exceeded". For all CommTool true positives, this reason accounts for a total of 36,531 alerts, while only accounting for 52 alerts generated uniquely by CommTool. This is because "Pages Per Session" is also a measure found in Arcane, and as such it does not provide a meaningful source of diversity between the tools.

1) Diversity based on bytes sent

Fig. 2 shows the number of true positives each tool had for ranges of bytes sent in HTTP requests. The value of bytes sent indicates the number of bytes that the response to the client had. We can see that there are three main clusters of activity when it comes to the tools' true positives. The first of these clusters corresponds to requests with responses within 0 to 8 kbytes. In this cluster we see that both Arcane and CommTool are active, with Arcane having an overall large number of unique alerts when compared to CommTool. This inverts only for requests whose response had between 4 and 6 kbytes, for which Arcane does not alert uniquely. The second cluster of activity concerns the range of 18 to 20 kbytes, where we see that both systems alert, but only CommTool alerts uniquely. Finally, the third cluster of activity is present between 30 to 36 kbytes. For this last cluster of activity, we see that only Arcane alerts with true positives. If we take a look at the same information for false positives (Fig. 3) we see that CommTool does have activity past the 20 kbytes mark, however, it only alerts with false positives.

After further discussions with the dataset provider we were told that neither CommTool nor Arcane make use of the number of bytes sent for a particular request. Nevertheless, this appears as a source of additional diversity between Arcane and CommTool. Because Arcane's output is based solely on HTTP access logs, Arcane's detection tends to improve as the duration of sessions increase. Over time, Arcane has access to more data and statistics that allow for a greater confidence in the alerts generated. This could help to explain why Arcane generates a higher number of alerts (both true and false positives) when compared to CommTool, as the number of bytes sent increases.

Taking a deeper look at the number of bytes sent in the requests, we show in Fig. 4 the average number of bytes sent for true positives over time. This information is shown for true positives generated by Arcane, only by Arcane, by CommTool and only by CommTool. What we see is that, for either tool, the average number of bytes sent is consistently higher for true positives they have uniquely alerted on when compared to all of their true positives (the ones they generated alerts for along with the other tool). This indicates that the alerting patterns of Arcane and CommTool appear to overlap for requests which generate responses with lower bytes, and as the number of bytes sent increases they diverge in their alerting behaviour.

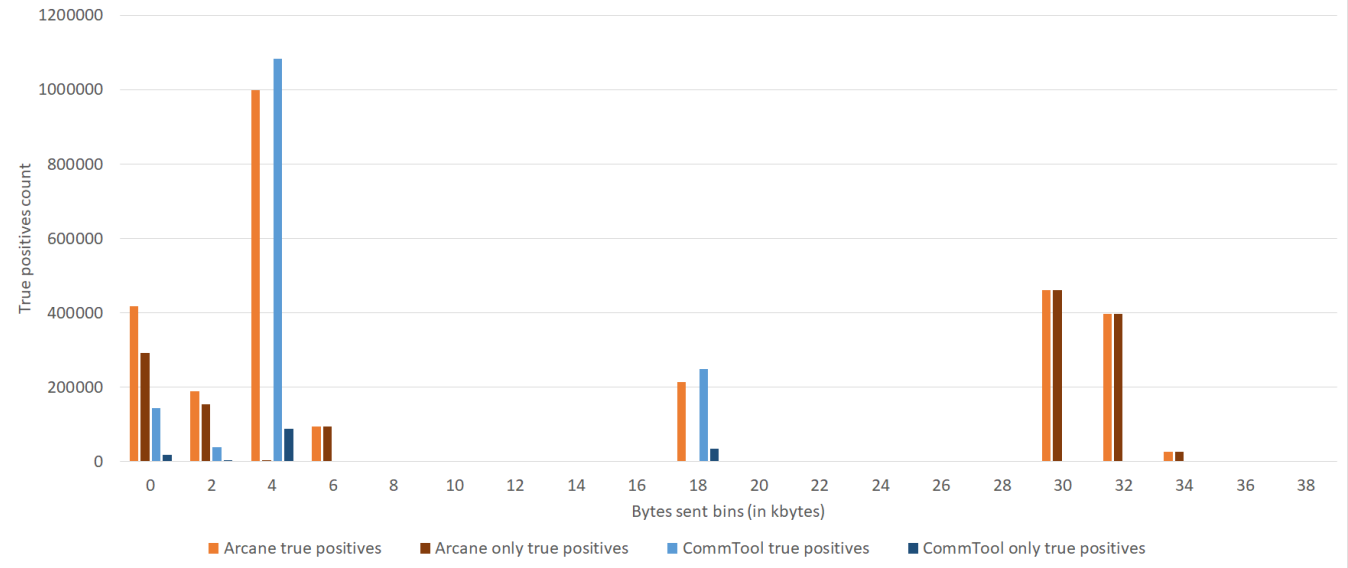

Fig. 2. True positives by bytes sent.

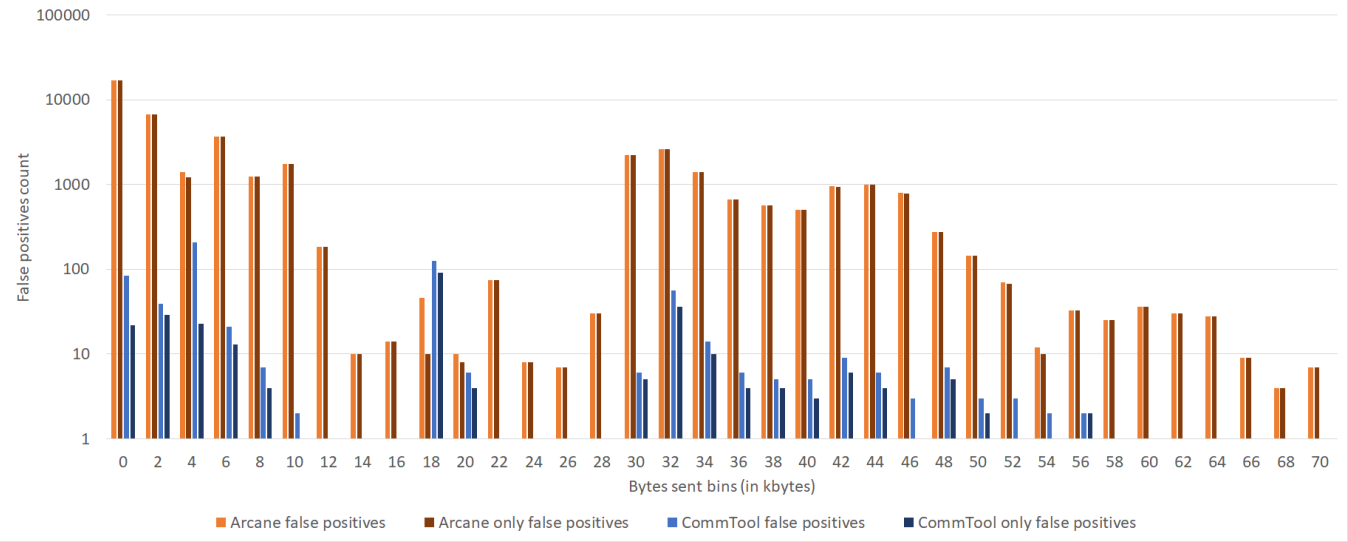

Fig. 3. False positives by bytes sent.

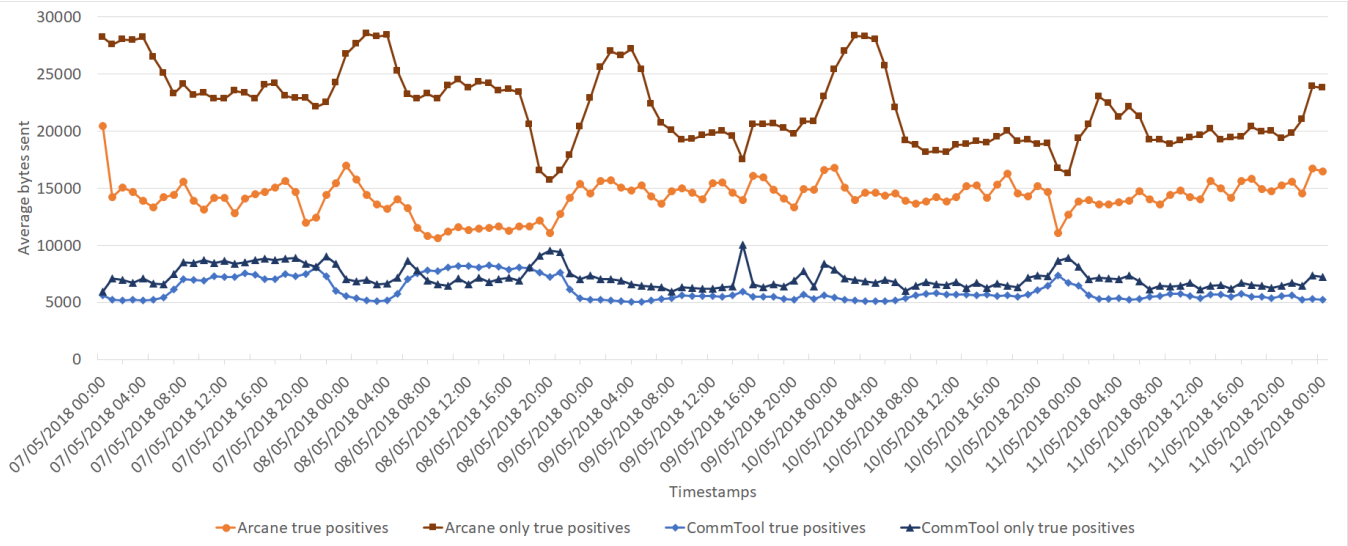

Fig. 4. Average bytes sent for true positives.

2) Diversity based on HTTP status

Another area in which Arcane and CommTool seem to differ in terms of their alerting patterns is based on the response status for HTTP requests. Table XI is a shortened version of the true positive counts generated by Arcane, Arcane-only, CommTool and CommTool-only based on HTTP response status. When considering that Arcane generates overall more alerts than CommTool, we can see that response statuses 302 and 405 are alerted differently by Arcane and CommTool. Investigating further the unique alerts generated by the tools we see that Arcane has a much higher number of unique alerts for HTTP response status 302 (Found), while CommTool is the only tool that generates unique alerts for status 405 (Method not found). Arcane does alert for requests whose response status was 405, however it does not do so uniquely, meaning that, for the case of HTTP response status 405, CommTool is presented as a *superset* of Arcane. From discussions with the dataset provider they confirmed that this is because CommTool sent a CAPTCHA at the start of the connection (with status 405). Because it was at the start of the connection (i.e. the CAPTCHA test was failed by the bot), CommTool detects it while Arcane does not.

TABLE XI. TRUE POSITIVE COUNTS BY HTTP RESPONSE STATUS (SHORTENED).

Status	**Arcane true positives**	**CommTool true positives**
200 (OK)	2,167,085	1,123,320
302 (Found)	418,115	143,578
405 (Method not allowed)	213,929	248,825
Status	**Arcane only true positives**	**CommTool only true positives**
200 (OK)	1,138,231	94,466
302 (Found)	291,710	17,173
405 (Method not allowed)	0	34,896

HTTP response status 302 (Found) indicates that the resource requested has temporarily been moved to another location, and the response returns a new URL that the client follows to find said resource. Status 405 (Method not allowed) indicates that the HTTP request method is not allowed for the specified resource. In the standard HTTP specification, the GET and HEAD methods should never return status 405, however, looking at the 34,896 true positives that only CommTool raised with this status, these are split in half between GET and POST (GET=18,750 and POST=16,146). CommTool-only alerts for status 405 are raised for two URI's, that being https://e-retail-domain.com (anonymised), a total of 18,717 times, and "https://e-retail-domain.com/fares.action" (anonymised), a total of 12,788. We note that CommTool has a specific rule to look at the latter of these URI's.

Fig. 5 shows the distribution over time for Arcane-only 302 true positives, and CommTool-only 405 true positives.

The fact that these are spread throughout the lifetime of the dataset indicates that this is a recurring difference between the tools, rather than a random occurrence.

3) Diversity based on user agent and unique IPs

One way to detect the presence of web scraping activities is to find evidence of botnets. This is because, as described previously, attackers often lack the resources to establish a centralized scraping system, and resort to the use of botnets to launch their scraping attacks. Evidence of botnet activity can be found by looking at the parameters sent with each HTTP request, for example, looking at the value of user agent. In HTTP requests, the value of user agent describes the software from where the request originated. Under normal circumstances, where a human performs a request, this value takes on the name or description of the browser that the user utilized. By looking at the value of user agent across multiple unique requests generated by different client IPs, we can correlate them in order to detect the presence of botnets. In Table XII we show the top user agents based on the number of total attack requests made. For each of them we show how many requests have been attacks, the number of unique IP addresses that have used the user agent for malicious requests, and the number of alerts generated by Arcane, Arcane-only, CommTool, and CommTool-only. What we see is that Arcane has a large number of unique alerts generated for the top 4 user agents (which have been used by more than 1,000 unique IP addresses). After these first 4, Arcane ceases to alert uniquely for any user agent. Meanwhile, CommTool has a lower number of unique alerts for the top 4 user agents, compared to Arcane, but from there on appears as a *superset* of Arcane (i.e. it detects all that Arcane does and more).

These results are aligned with the way both tools work. Arcane is primarily server-side, therefore it has the ability to look at multiple HTTP requests and correlate them together to find botnets. As such, it is not surprising that it performs better for user agents which have been used by more unique IP addresses, as these potentially constitute evidence of larger botnets. CommTool, being a client-side tool, cannot look at multiple requests in order to correlate them, instead analysing each connection individually.

We calculated the sensitivity and specificity rates that Arcane and CommTool have for each individual user agent. These are presented in Table XIII for the top 4 user agents (the only user agents which had unique true positives from both tools).What we see is that, as the number of unique IPs using a specific user agent decreases, CommTool's sensitivity (true positives) increases. The same appears to be true for CommTool's and Arcane's specificity.

Note that, for user agent (D), specificity cannot be calculated due to the fact that all HTTP sessions with that user agent were labelled as attacks.

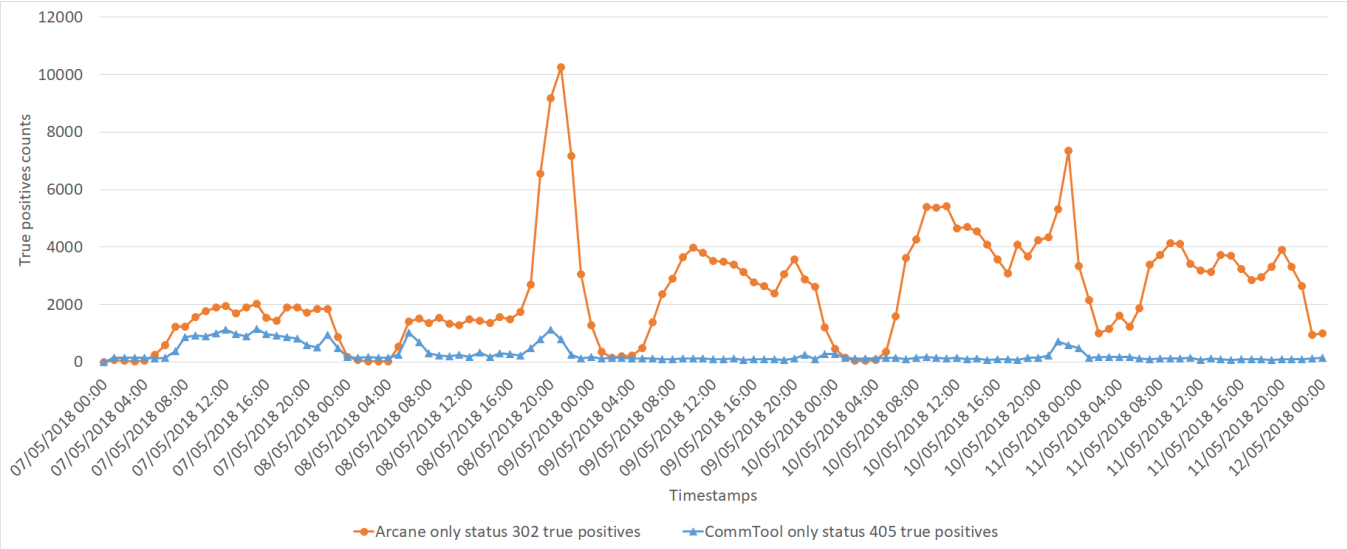

Fig. 5. Arcane only 302 and CommTool only 405 true positives.

TABLE XII. TRUE POSITIVES BY USER AGENT (SHORTENED).

User agent	Total attacks	Unique IP addresses	Arcane true positives	CommTool true positives	Arcane only true positives	CommTool only true positives
(A) - Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.116 Safari/537.36 Edge/15.15063	2,775,396	162,149	983,747	537,247	540,126	93,626
(B) - Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; rv:11.0) like Gecko	1,109,404	21,195	1,037,946	469,769	581,221	13,044
(C) - Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko	705,335	14,783	550,182	293,541	292,321	35,680
(D) - Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/61.0.3163.100 Safari/537.36	230,682	1,053	226,742	211,528	16,532	1,318
(E) - Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)	789	297	590	789	0	199
(F) - Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.116 Safari/537.36	610	313	23	610	0	587
(G) - Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.116 Safari/537.36	559	271	11	559	0	548
(H) - Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:49.0) Gecko/20100101 Firefox/49.0	505	269	16	505	0	489
(I) - Mozilla/5.0 (iPhone; CPU iPhone OS 11_3 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/11.0 Mobile/15E148 Safari/604.1	169	42	34	169	0	135
(J) - Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/59.0.3071.115 Safari/537.36	56	28	0	56	0	56

TABLE XIII. ARCANE AND COMMTOOL SENSITIVITY AND SPECIFICITY FOR SPECIFIC USER AGENTS

User agent		Arcane	CommTool
(A)	**Sensitivity**	35.45%	19.36%
	Specificity	38.39%	52.57%
(B)	**Sensitivity**	93.56%	42.34%
	Specificity	83.18%	98.88%
(C)	**Sensitivity**	78.00%	41.62%
	Specificity	91.02%	99.50%
(D)	**Sensitivity**	98.29%	91.70%
	Specificity	N/A	N/A

By only taking into account Arcane's alerts for the top 4 user agents we can substantially improve the specificity of both Arcane by itself (78.51% to 99.81%) and 1oo2 adjudication system (78.37% to 99.60%), while keeping the sensitivity of the 1oo2 the same (at 61.07%) as shown in Table XIV (cf. Table V and VIII for comparison).

TABLE XIV. ARCANE TOP 4 USER AGENTS AND COMMTOOL ALL USER AGENTS SENSTIVITY AND SPECIFICITY.

	Sensitivity	Specificity
Arcane	58.002%	99.81%
CommTool	31.42%	99.69%
1oo2	61.07%	99.60%
2oo2	28.36%	99.91%

These results indicate that Arcane only brings benefits when monitoring user agents used by more than 1000 unique IP addresses. For any user agent with less than 1000 addresses, Arcane provides no extra benefits in true positives, and yet provides a lot more false positives.

In Fig. 6 we show the sensitivity rates of each tool as well as the 1oo2 and 2oo2 configurations when analysing requests coming from user agents used by less than "x" unique IPs. What we observe is that, for user agents used by less than

1000 unique IPs, CommTool has a perfect sensitivity value, while Arcane has a sensitivity rate close to 10%. However, when analysing user agents employed by more than 1000 unique IPs, Arcane appears as a more sensitivity system, when compared to CommTool.

We note that, in the dataset we have analysed, there were no instances of user agents used by more than 1000 unique IP addresses but less than 14000. This means we cannot correctly identify with precision, the point at which both Arcane and CommTool have a drop in their sensitivity rates.

In Fig. 7 and 8 we show the specificity rates of each tool in the same manner. Arcane (and consequently 1oo2) have the biggest change in specificity based on the number of unique IPs using the same user agent, with an increase of roughly 0.02 as the number of unique IPs using the same user agent increases. For CommTool (and consequently 2oo2), there is a slight decrease of roughly 0.001 as the number of unique IPs increases.

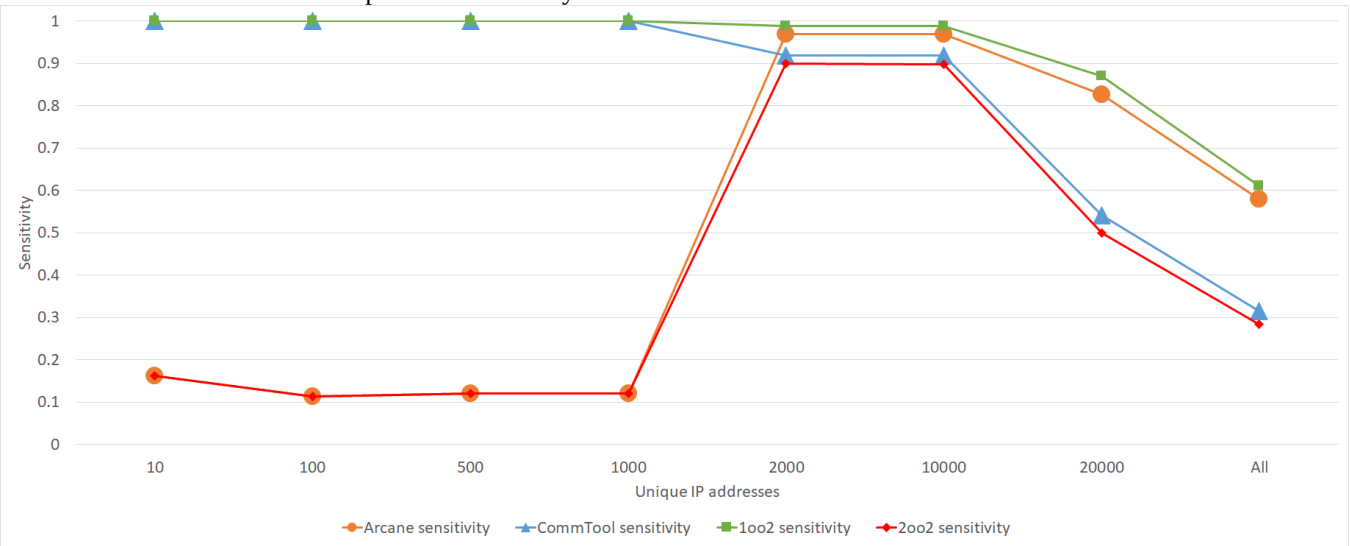

Fig. 6. Sensitivity rates when analysing user agents used by less than "x" unique IPs.

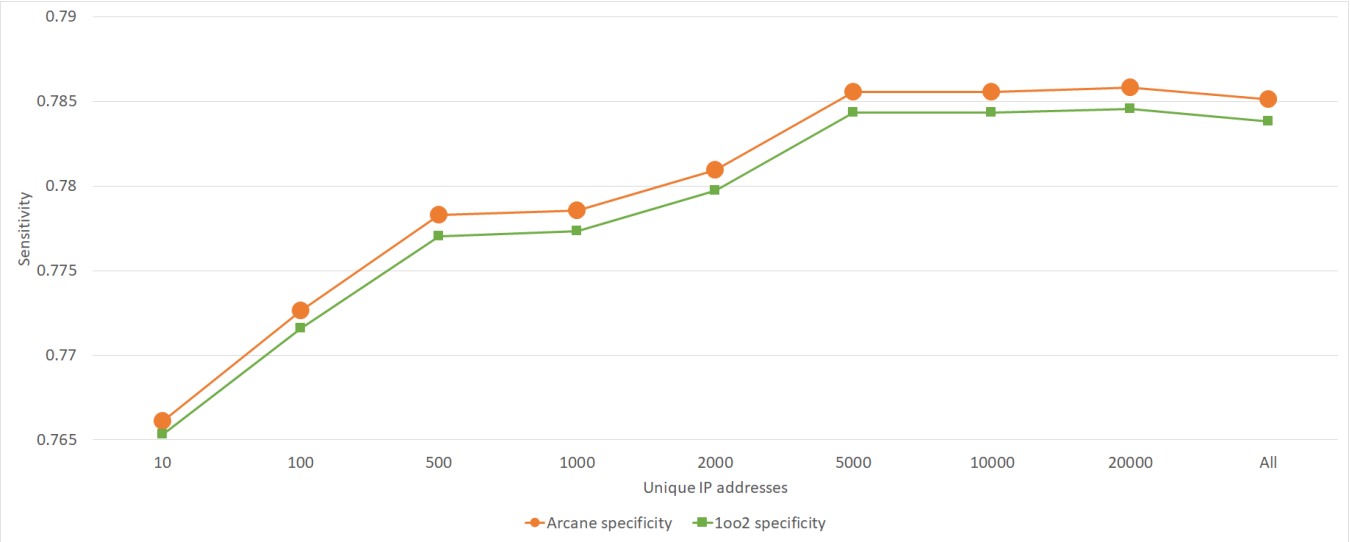

Fig. 7. Specificity rates when analysing user agents used by less than "x" unique IPs (Arcane and 1oo2).

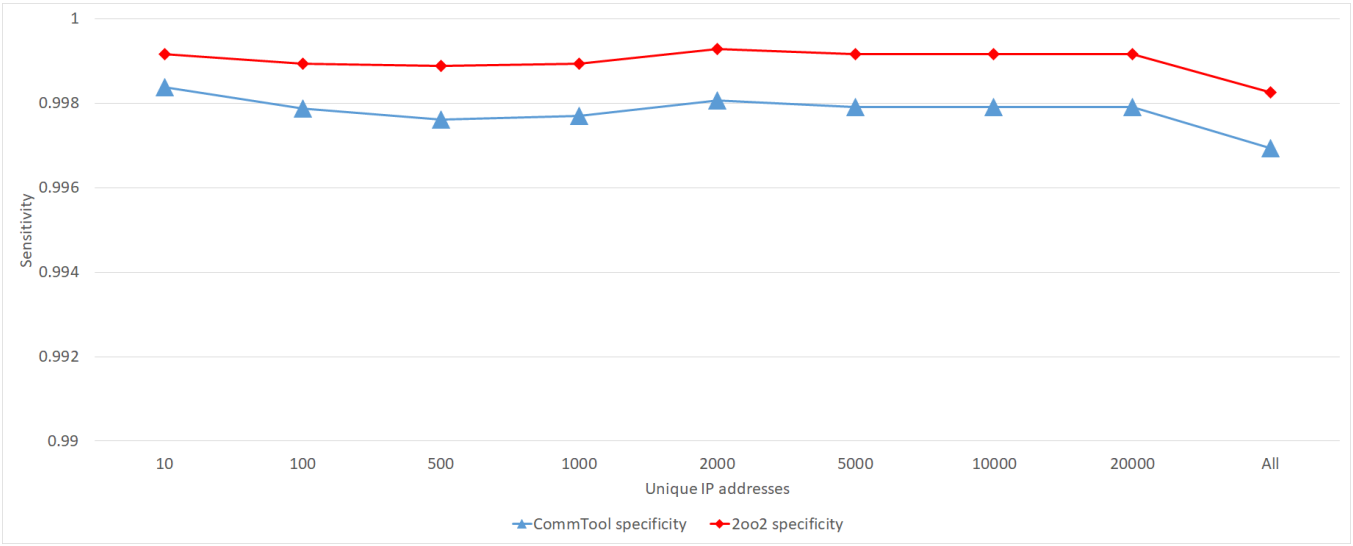

Fig. 8. Specificity rates when analysing user agents used by less than "x" unique IPs (CommTool and 2oo2).

D. *Diversity in the geolocation of the IP addresses*

Figure 9 below shows where the majority of malicious web scraping requests were made from. It is possible to see that they mostly originate from central Europe. The distribution of true positives generated by both Arcane and CommTool looks identical to that of all malicious web scraping requests, where both tools generally raised alerts for central Europe.

Figures 10 and 11 show the distribution of false positives generated by Arcane-only and CommTool-only. In this case we see a significant difference in the alerting patterns. Arcane-only false positives mostly originate from central Europe once again, but CommTool-only false positives appear mostly to be coming from Mexico.

Fig. 9. Geolocation of all web scraping requests.

Fig. 10. Geolocation of Arcane-only false positives.

Fig. 11. Geolocation of CommTool-only false positives.

This difference in alerting patterns for the false positives of either tool likely comes down to the use of different IP reputation databases. Because the database used by Arcane is built and maintained by the dataset provider, it stands to reason that it would contain more IPs and networks from central Europe, seeing as most malicious requests observed on their networks are coming from this area. For the case of CommTool however, the database of known violators is a global one, and as such, this pattern differs.

V. Discussion, Conclusions and Further Work

Our results show clear signs of diversity benefits, as evidenced by the values of sensitivity and specificity for 1oo2 and 2oo2 adjudication systems. There are two major factors for why we see differing alerting patterns between Arcane and CommTool:

- Firstly, the mode of operation between the two tools is substantially different from one another. CommTool works primarily on the client-side, and most of its detection capabilities are due to JavaScript tests run on the client's device. This translates to a generally faster decision as to whether a request is malicious or not but prevents CommTool from correlating between various different connections. Arcane is exclusively server-side, and thus is capable of looking and correlating between multiple different client requests. This means that Arcane can more easily detect the presence of botnets. It also means that Arcane outputs more accurate alerts for longer connections.

- Secondly, the use of different known violator databases leads both tools to detect different connections. This, however, is not inherent to their operation modes.

There are also general configuration changes between Arcane and CommTool that make them alert differently. Internally, Arcane is used only to analyse HTTP access logs a posteriori, without taking actions upon them. This is the case because Arcane is configured more loosely, so as to generate more alerts overall. Because these alerts are not acted upon automatically, the company that provided the dataset can afford to generate more false positives, to increase the probability of increasing the true positive rate.

Based on the results of our study and the observations we made through the paper, we derived some practical recommendations on the diverse deployment of these tools:

- Arcane provides value for this application only when monitoring user agents used by more than 1000 IP addresses. For user agents with less than 1000 IP addresses, all true positives from Arcane are also true positives in CommTool, while Arcane also generates a large number of false positives. For this application it seems the SOC operator can ignore alerts generated by Arcane-only for user agents with less than 1000 IP addresses, as they are highly likely to be false positives.
- The geolocation of the malicious web scrapers for this application seems to be primarily in Europe. It seems that IP addresses that are not from Europe and reported as such by CommTool are more likely to be false positives for this application. It will be interesting to see whether this also holds for other applications that are monitored by the dataset provider.
- Even though the dataset provider confirmed that neither CommTool nor Arcane make use of the number of bytes sent for a particular request when making a decision on whether to alert or not, it appears that the alerting patterns of Arcane and CommTool appear to overlap for requests which generate responses with lower bytes, and as the number of bytes sent increases they diverge in their alerting behaviour. Because Arcane's output is based solely on HTTP access logs, Arcane's detection tends to improve as the duration of sessions increase. Arcane has access to more data and statistics that allow for a greater confidence in the alerts generated over time. This helps to explain why Arcane generates a higher number of alerts (both true and false positives) when compared to CommTool, as the number of bytes sent increases. Hence bytes sent seems to be an interesting parameter to consider when considering changes to the configurations of these tools

The company that provided the dataset plan to reconfigure the defences based on the findings and recommendations made here. We will then do a follow up study to check if the sensitivity and specificity of the systems (including the diverse combinations) has improved in the face of a changing threat profile.

We note that the results presented here are only prima facie evidence of the usefulness of diversity. More analysis with more datasets for more applications would enable us to get more confidence on whether the diversity observed here is indeed observed more generally in other applications. To this end current and further work is to analyse data from more applications to analyse whether what we have observed here is a typical behaviour of these tools, or whether they differ markedly depending on the environment and the nature of the applications monitored.

ACKNOWLEDGMENT

This work is supported by the European Commission through the H2020 programme under grant agreement 700692 (DiSIEM) and by the UK EPSRC project D3S.

REFERENCES

[1] D. Stevanovic, A. An and N. Vlajic, "Feature evaluation for web crawler detection with data mining techniques", *Expect Systems with Applications*, vol. 39, no. 10, pp. 8707-8717, 2012.

[2] A. Stassopoulou and M. D. Dikaiakos, "Web robot detection: a probabilistic reasoning approach", *Computer Networks*, vol. 53, no. 3, pp. 265-278, 2009.

[3] A. Al-Bataineh and G. White, "Analysis and detection of malicious data exfiltration in web traffic", in *7th International Conference on Malicious and Unwanted Software*, Fajardo, PR, USA, 2012.

[4] B. Littlewood and L. Strigini, "Redundancy and diversity in security", *Computer Security Esorics*, 3193, pp. 423-438, 2004.

[5] M. Garcia, A. Bessani, I. Gashi, N. Neves, R. Obelheiro, "Analysis of operating system diversity for intrusion tolerance", *Software – Practice & Experience*, vol. 44, no. 6, pp. 735-770, 2014.

[6] S. Singh, M. Cukier and W.H. Sanders, "Probabilistic validation of an intrusion-tolerant replication system", in *2003 International Conference on Dependable Systems and Networks*, San Francisco, CA, USA, 2003.

[7] V. Gupta, V. Lam, H.V. Ramasamy, W. H. Sanders, S. Singh, "Dependability and performance evaluation of intrusion-tolerant server architectures, *Lecture Notes in Computer Science*, vol. 2847, pp. 81-101, 2003.

[8] P. Bishop, R. Bloomfield, I. Gashi and V. Stankovic, "Diversity for security: a study with off-the-shelf antivirus engines", in *IEEE 22nd International Symposium on Software Reliability Engineering*, Hiroshima, Japan, 2011.

[9] P. Popov, A. Povyakalo, V. Stankovic and L. Strigini, "Software diversity as a measure for reducing development risk", in *Tenth European Dependable Computing Conference*, Newcastle upon Tyne, UK, 2014.

[10] B. Littlewood and D. R. Miller, "Conceptual modeling of coincident failures in multiversion software", IEEE Transactions on software engineering, vol. 15, no. 12, pp. 1596-1614, 989.

[11] P. Cunningham and J. Carney, "Diversity versus quality in classification ensembles based on feature selection", in European Conference on Machine Learning, 2000, pp. 109-116: Springer.

[12] L. I. Kincheva and C. J. Whitaker, "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy", Machine learning, vol. 51, no. 2, pp. 181-207, 2003.

[13] D. E. Eckhardt, L. D. Lee, "A theoretical basis for the analysis of multiversion software subject to coincident errors", IEEE Trans. Software Eng., vol. SE-11, no. 12, pp. 1511-1517, 1985.

[14] A. Avižienis, L. Chen, "On the implementation of n-version programming for software fault-tolerance during program execution", Proc. COMPSAC 77, pp. 149-155, 1977.