A First Look at HTTP(S) Intrusion Detection using NetFlow/IPFIX

Olivier van der Toorn, Rick Hofstede, Mattijs Jonker, Anna Sperotto
Design and Analysis of Communication Systems (DACS)
Centre for Telematics and Information Technology (CTIT)
University of Twente, Enschede, The Netherlands
E-mail: o.i.vdtoorn@student.utwente.nl, {r.j.hofstede, m.jonker, a.sperotto}@utwente.nl

***Abstract*—Brute-force attacks against Web site are a common area of concern, both for Web site owners and hosters. This is mainly due to the impact of potential compromises resulting therefrom, and the increased load on the underlying infrastructure. The latter may even result in a Denial-of-Service (DoS). Detecting brute-force attacks – and ultimately mitigating them – is therefore of great importance. In this paper, we take the first step in this direction, by presenting a network-based approach for detecting HTTP(S) dictionary attacks using NetFlow/IPFIX. We have developed a prototype Intrusion Detection System (IDS), released as open-source software, by means of which we can achieve accuracies close to 100%.**

***Index Terms*—Network security, Intrusion detection, NetFlow/IPFIX.**

I. Introduction

Wordpress, Joomla and Drupal have become the dominating Content Management Systems (CMSes) with a market share of almost 30% that is increasing evermore [1]. The problem with many CMSes is that they are poorly protected against brute-force attacks, which makes them a prime attack target [2], [3]. The simple means by means of which Web applications can be built attracts also less-technical people, who are often completely unaware of security risks. Besides CMS backends, there is another common attack target on Web servers that has existed for much longer than CMSes: HTTP Basic Authentication (BA). HTTP BA is a standardized part of HTTP and allows for password-protecting directories and files.

Antagonist, one of the major Web hosting companies in the Netherlands that hosts over 100k Web applications, faces brute-force attacks against HTTP BA and CMS backends in a non-stop fashion. The problem with these attacks, typically referred to as *dictionary attacks*,[1] is twofold. On one hand, site owners have the risk of having their Web site compromised, after which the system may be misused for illegal activities, such as distributing illegal content or sending SPAM. On the other hand, dictionary attacks may constitute a Denial-of-Service attack on the hosting infrastructure, because login pages, which are typically PHP-based, have to be processed with every request.

Dictionary attacks against Web applications can be easily detected by means of access logs. This host-based approach is however hardly scalable in larger networks, as access to individual machines is required. We therefore take a flow-based approach in this work, given the many advantages provided by flow export technologies [4].

To the best of our knowledge, no work currently exists on the flow-based detection of brute-force attacks against Web sites. Work that comes close to ours is presented in [5], but it works at the level of individual packets, rather than flows. Other related works are presented in [6], which cover the flow-based detection of dictionary attacks against SSH daemons. Since there are many commonalities among dictionary attacks against SSH daemons and Web servers, we will reuse parts of the findings and methodology presented in related works.

The goal of this paper is to investigate the patterns of HTTP(S) dictionary attacks at the flow-level, and define and validate the corresponding signatures. We describe and analyze the attack patterns that are common for HTTP(S) dictionary attacks in Section II and III, respectively. After that, we present the validation in Section IV and discuss our conclusions and future work in Section VI.

II. Attack Types & Phases

There are three common types of authentication mechanisms by means of which Web sites are protected. Brute-force attacks over HTTP(S) usually target one of these mechanisms. These mechanisms are as follows:

- **HTTP Basic Authentication (HTTP BA)** – HTTP BA is an authentication mechanism that provides password protection on the level of files and directories. It uses a dedicated window that is not embedded in a Web page, prompting for credentials. Once authenticated, the Web server sends the requested object; otherwise, a *401 Unauthorized* status code is returned.
- **Form-based Authentication (FA)** – FA uses embedded Web forms that allow credentials to be submitted to the Web server using a HTTP POST request. If the credentials are valid, a session is set up for the client.
- **XML-RPC** – XML-RPC provides a means for handling Remote Procedure Calls (RPCs). RPCs have been used in dictionary and DDoS attacks recently [7], [8]. Authentication attempts consist of XML files with the RPC's name, together with credentials.

[1] Attacks that rely on lists with frequently used login credentials, commonly referred to as *dictionaries*.

TABLE I: Overview of CMS authentication mechanisms

CMS	URL	Attributes
Wordpress	/wp-login.php	username, password
	/xmlrpc.php	username, password
Joomla	/administrator/index.php ⊢ ?option=com_login	username, password access token, task
Drupal	/?q=user /?q=user/login	username, password access token, task
	/xmlrpc.php	username, password

The authentication mechanisms of the aforementioned CMSes are listed in Table I. None of the CMSes use HTTP BA by default. Noteworthy about the FA authentication mechanism for Wordpress is that it merely relies on usernames and passwords, while both Joomla and Drupal require an additional *access token* to authenticate to the backend. This access token is randomly generated when the login page is processed and, as a consequence, a user must go through the original login page to obtain a valid access token. Joomla and Drupal also use an attribute for specifying which action needs to be taken upon submission of the form, i.e., the *task* attribute.

What attacks against HTTP BA, FA and XML-RPC have in common is that they are typically dictionary-based. As such, they are brute-force in nature and consist of authentication attempts using frequently-used credentials in an automated fashion. While no work has been performed yet on the detection of such attacks at the network-level, the detection of similar attacks against SSH daemons has been studied extensively. These works identify three phases in dictionary attacks [6]:

- **Scan** – Attackers scan for specific ports with listening daemons on servers.
- **Brute-force** – Attackers try to authenticate with varying credentials.
- **Compromise** – Attackers have gained access to the targeted service by using correct login credentials. In the specific case of HTTP(S) attacks, this often means that attackers have gained access to the backend of a Web application, allowing the attacker to send SPAM or upload illegal content, for example.

Although dictionary attacks over HTTP(S) typically consist of these phases as well, the *scan* phase is slightly different from the definition given in [6]. Dictionary attack tools need target URLs instead of IP addresses, because multiple Web sites can be hosted on the same Web server using *virtual hosts*. Based on the URL, a request is dispatched to the applicable *virtual host*. Scanning for Web servers merely results in a list of IP addresses, which have to be translated to one or more URLs. To this end, an attacker may use reverse DNS lookups or services such as the Hurricane Electric BGP Toolkit.

III. Attack Tool Behavior & Signatures

To study the network-level behavior of attacks against Web servers, we have acquired dictionary attack tools through previous research experience and by searching the Web. We selected Hydra v8.1-pre, Patator v0.7-beta and Medusa v2.1.1, which support dictionary attacks against HTTP BA and FA. In addition, we selected HTTP-Brute v0.2, an NMAP extension, because its documentation suggests that HTTP pipelining is supported.[2] We also selected Intrinsec XML-RPC Scanner v0.4 and XmlRpcBrute v1.0 for attacking XML-RPC, which we found to be the only tools supporting such attacks. Using the aforementioned tools, we have performed dictionary attacks upon against all three authentication mechanisms in a lab environment, while capturing the generated traffic.

Based on the analysis of the listed attack tools and the impact of SSL on the resulting network traffic, we have created signatures for dictionary attacks over both HTTP and HTTPS. These signatures are listed in Table II, and based on the upper and lower limits of the number of packets and bytes per flow for each individual tool. To create the HTTPS signatures, all combinations of SSLv3/TLSv1.x with AES, (3)DES and RC4 encryption were used, whenever possible. Per such combination, commonly first-picked algorithms for data authentication (HMAC) and key exchange were used, and always without compression. While varying such algorithms will affect the signatures both in terms of bytes and number of packets, this was not done as part of this work for the sake of brevity.

Several observations can be made regarding the signatures in Table II. First, since a minimum of five packets is necessary for a login attempt over HTTP, and seven for HTTPS, signature bounds with a lower number of PPF are not realistic. Notwithstanding, these occur for HTTP-Brute, and in some cases Hydra. Investigation has shown that in these cases a few outlying flows push boundaries down, while consisting only of zero-payload TCP packets. For HTTP-Brute, this behavior results from an up front port scan triggered by the NMAP API. For Hydra, this is because connections are sometimes closed ungracefully. Given these findings, we ignore the outlying flows in the definition of the signatures. Second, the use of SSL does not change the characteristics of the attack tools, as they treat an SSL socket in the same way as a TCP socket. As such, the overhead introduced by the use of SSL is constant, which is also related to the fact that attack flows are rather small in terms of packets and bytes. Last, we can observe a clear increase in the number of BPF for attacks against XML-RPC, which can be accounted to the use of XML files.

IV. Validation

In this section, we validate the signatures presented in the previous section. To do so, we have developed an open-source IDS prototype[3] to automate the detection of HTTP(S) dictionary attacks based on the presented signatures. We start this section by discussing the dataset we have used to validate the signatures. After that, we show and elaborate on the validation results in terms of accuracy.

[2]We found that while NMAP's HTTP library supports request pipelining, this functionality is not actually being used by HTTP-Brute.

[3]The prototype is available at https://github.com/ut-dacs/https-ids.

TABLE II: Attack signatures based on PPF and BPF

		PPF				*BPF*			
	Attack tool	**HTTP**	**HTTPS**	**Signature HTTP**	**Signature HTTPS**	**HTTP**	**HTTPS**	**Signature HTTP**	**Signature HTTPS**
BA	HTTP-Brute	3 – 6	3 – 12	5 – 6	7 – 12	132 – 355	132 – 1425	367 – 438	789 – 1683
	Hydra	5 – 6	7 – 9			372 – 432	789 – 1105		
	Medusa	6	8 – 10			395 – 438	952 – 1328		
	Patator	6	9 – 10			367 – 379	1091 – 1683		
FA	Hydra – Drupal	2 – 9	4 – 12	5 – 12	8 – 17	100 – 696	216 – 1365	363 – 1130	828 – 2885
	Medusa – Drupal	5 – 9	8 – 12			560 – 736	1169 – 1616		
	Patator – Drupal	8 – 12	12 – 16			686 – 848	1528 – 2328		
	Hydra – Joomla	5 – 9	2 – 10			371 – 674	100 – 1349		
	Patator – Joomla	9 – 12	13 – 17			1007 – 1130	1941 – 2885		
	Hydra – Wordpress	5 – 6	8 – 10			363 – 553	828 – 1281		
	Medusa – Wordpress	5 – 8	8 – 10			480 – 607	1089 – 1432		
	Patator – Wordpress	6 – 7	10 – 12			434 – 481	1179 – 1787		
XML RPC	Intrinsec XML-RPC Scanner	5 – 6	7 – 8	5 – 6	7 – 8	770 – 826	1105 – 1433	770 – 889	1105 – 1433
	XmlRpcBrute	5 – 6	▨		▨	823 – 889	▨		▨

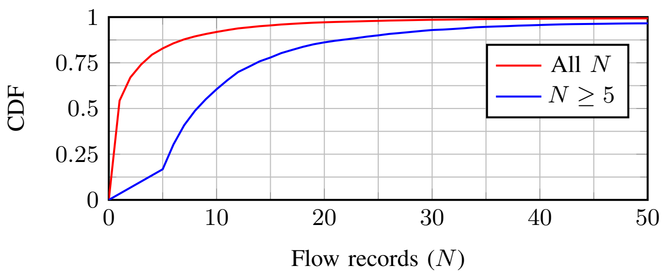

Fig. 1: Distribution of consecutive flow records with similar authentication URLs.

A. Dataset

For the validation of this work, we have created a month-long dataset of flow data exported using IPFIX, captured over the course of June/July 2014. The dataset consists of traffic towards a Web server hosting roughly 100 Web sites, mainly built using Wordpress and Joomla, and features 4.7M flow records, worth of 179.7GB and 205.8M packets. Over the full dataset, a total of 2.2G requests have been performed to the Web server, resulting in 71.7k requests on average per day. It should be noted that the exported flow data includes URLs and HTTP status codes of HTTP traffic as well, serving as the ground-truth for the validation. Since neither status codes nor (full) URLs can be exported for HTTPS traffic (due to its encrypted nature), we limit our validation to HTTP traffic.

Since the dataset consists of raw flow data only, it must be post-processed by aggregating consecutive flow records into attacks. In the case of FA and XML-RPC attacks, we perform the aggregation based on similar[4] consecutive backend URLs, as listed in Table I. Such aggregation would fail for HTTP BA, since it does not use a fixed set of URLs as the other mechanisms do, since any file on a Web server can be HTTP BA-protected. Instead, we aggregate flow records based on both URLs and HTTP status codes; N consecutive flow records featuring identical URLs and a *401 Unauthorized* status code are aggregated into attacks.

[4]As we have seen cases where attack tools append parameters to the URLs, we use wildcards after the URLs in Table I.

After post-processing, we can perform a one-to-one comparison between the detection results and the ground-truth. In total, we observed 3 attacks against HTTP BA, 1153 attacks against FA and 3120 attacks against XML-RPC.

B. Detection Accuracy

In the remainder of this section, we distinguish between *attacks* and *tuples*. *Attacks* always feature a single attacker and one or more targets. Every pair of attacker and target is referred to as a *tuple*. As a consequence, every attack consists of one or more tuples. Since our ground-truth consists of data of only a single machine, we perform the validation in terms of tuples, where every tuple thus consists of an attacker and the monitored machine as a target. A tuple can only be present as part of an attack in the ground-truth if N or more consecutive flow records with similar backend URLs have been found.

To evaluate the detection accuracy, we use the typical set of metrics for evaluating the performance of IDSes, consisting of True Positive (TP), False Positive (FP), True Negative (TN), False Negative (FN) and Accuracy (Acc) [6]. This is done in two dimensions. First, we consider multiple values of the *flow record threshold* (N), based on the distribution of attack sizes in terms of similar consecutive flow records, as shown in Fig. 1. To cover 25%, 50%, 75% and 95% of the attacks that feature at least 5 flow records, we take thresholds of 6, 9, 14 and 37 flow records, respectively. Second, we evaluate the detection accuracy when using only the PPF values, BPF values, and both PPF and BPF values of our signatures. The results are listed in Table III. Note that we express the results in terms of percentages of the previously defined evaluation metrics. For example, the *True Positive Rate* (TPR) is the

TABLE III: Detection accuracy

	Flow record threshold	TPR	TNR	FPR	FNR	*Acc*
PPF	6	0.948	0.947	0.053	0.052	0.947
	9	0.936	0.969	0.031	0.064	0.969
	14	0.923	0.983	0.017	0.077	0.983
	37	0.918	0.996	0.004	0.082	0.996
BPF	6	0.970	0.958	0.042	0.030	0.958
	9	0.962	0.974	0.026	0.038	0.973
	14	0.942	0.984	0.016	0.058	0.984
	37	0.878	0.996	0.004	0.122	0.995
PPF+BPF	6	0.937	0.965	0.035	0.063	0.965
	9	0.920	0.979	0.021	0.080	0.979
	14	0.899	0.989	0.011	0.101	0.988
	37	0.864	0.997	0.003	0.136	0.997

percentage of correctly identified tuples for which N or more consecutive flow records with similar CMS backend URLs are reported in the ground-truth.

Increasing the flow record threshold generally results in a higher accuracy. It can be observed that the TPR and FPR decrease slightly, with an increasing flow record threshold, while the TNR and FNR increase slightly. Another observation that can be made from the validation results is that only using the number of PPF of the signatures yields best results at higher flow record thresholds, while the BPF approach yields best results at lower thresholds. Combining the number of PPF and BPF yields the highest accuracies.

Due to the encrypted nature of HTTPS traffic, we cannot validate the HTTPS attack signatures by comparing detection results to our ground-truth. However, given that the corresponding signatures have been derived in the same way as for attacks over HTTP, we believe that the detection results for attacks over HTTPS are similar to those for attacks over HTTP.

V. Discussion

The dataset used for validation of this work, as described in Section IV-A, consists of flow data exported using IPFIX. This data features L5 information, i.e., URLs in HTTP traffic, besides the traditional L3 and L4 information typically exported in flow data. Based on the per-flow information on URLs, we could have performed HTTP intrusion detection without the need of generating signatures, as shown in Table II. That approach would also have avoided false detections, as benign traffic could be discriminated based on the URL. However, given that the amount of Web traffic is rapidly increasing in favor of HTTPS, detection based on URLs is no longer feasible, as (full) URLs cannot be extracted from traffic anymore due to end-to-end encryption. This advocates the use of signatures that are resilient against encryption, as presented in this work. In addition, the fact that our approach does not rely on L5 information makes it widely deployable, as any data exported using NetFlow or IPFIX can be used.

VI. Conclusions

This paper has presented the first work on the flow-based detection of HTTP(S) dictionary attacks. By means of analyzing attack tools, defining flow-level fingerprints and developing a prototype IDS, we have achieved promising results in detecting attacks against HTTP(S) BA, FA and XML-RPC. Validation of our signatures has shown that HTTP(S) dictionary attacks are identified accurately and that signatures that utilize both the PPF and BPF yield the best detection results. However, there are false positives, which are mainly caused by automated yet legitimate traffic, such as Web crawlers and calendar fetchers. We realize that this traffic can be of great importance to Web site owners, as they often rely on search engine rankings for their income, for example. Further investigation of this traffic will therefore be part of our future work.

In talks with Antagonist, we have learned that a system as presented in this paper may prove very useful. For example, it could be integrated with an automated system for blocking attackers based on detection results of our IDS. Requests from blocked hosts could be forwarded to a static landing page, from which one can choose to be unblocked. Since such behavior is not understood by the current generation of attack tools, humans can easily be unblocked while automated attacks are effectively mitigated.

Acknowledgments

Special thanks go to Antagonist B.V. for their valuable contributions. This work was partly funded by FLAMINGO, a Network of Excellence project (ICT-318488), and SALUS, a STREP project (ICT-313296), both supported by the European Commission under its Seventh Framework Programme.

References

[1] W3Techs, "Historical yearly trends in the usage of content management systems for websites," June 2014, accessed on 21 January 2015. [Online]. Available: http://w3techs.com/technologies/history_overview/content_management/all/y

[2] M. Mimoso, "Hackers Using Brute-Force Attacks to Harvest WordPress Sites," April 2013, accessed on 21 January 2015. [Online]. Available: http://threatpost.com/hackers-using-brute-force-attacks-harvest-wordpress-sites-041513/77730

[3] T. Perez, "Understanding Denial of Service and Brute Force Attacks - WordPress, Joomla, Drupal, vBulletin," March 2014, accessed on 21 January 2015. [Online]. Available: http://blog.sucuri.net/2014/03/understanding-denial-of-service-and-brute-force-attacks-wordpress-joomla-drupal-vbulletin.html

[4] R. Hofstede, P. Celeda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow Monitoring Explained: From Packet Capture to Data Analysis with Netflow and IPFIX," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 2037–2064, 2014.

[5] R. Koch, "Systemarchitektur zur Ein- und Ausbruchserkennung in verschlsselten Umgebungen," Ph.D. dissertation, Universität der Bundeswehr München, Germany, 2011.

[6] R. Hofstede, L. Hendriks, A. Sperotto, and A. Pras, "SSH Compromise Detection using NetFlow/IPFIX," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 20–26, 2014.

[7] D. Cid, "New Brute Force Attacks Exploiting XMLRPC in WordPress," July 2014, accessed on 21 January 2015. [Online]. Available: http://blog.sucuri.net/2014/07/new-brute-force-attacks-exploiting-xmlrpc-in-wordpress.html

[8] I. Zeifman, "New WordPress and Drupal Denial Of Service Vulnerability Fixed," Incapsula, August 2014, accessed on 21 January 2015. [Online]. Available: http://www.incapsula.com/blog/new-vulnerability-xmp-rpc-wp-drupal-fixed.html