Attacks Landscape in the Dark Side of the Web

Onur Catakoglu
Eurecom & Monaco Digital Security Agency
catakogl@eurecom.fr

Marco Balduzzi
Trend Micro Research
marco_balduzzi@trendimicro.com

Davide Balzarotti
Eurecom
davide.balzarotti@eurecom.fr

ABSTRACT

The Dark Web is known as the part of the Internet operated by decentralized and anonymous-preserving protocols like Tor. To date, the research community has focused on understanding the size and characteristics of the Dark Web and the services and goods that are offered in its underground markets. However, little is still known about the *attacks landscape* in the Dark Web.

For the traditional Web, it is now well understood how websites are exploited, as well as the important role played by Google Dorks and automated attack bots to form some sort of "background attack noise" to which public websites are exposed.

This paper tries to understand if these basic concepts and components have a parallel in the Dark Web. In particular, by deploying a high interaction honeypot in the Tor network for a period of seven months, we conducted a measurement study of the type of attacks and of the attackers behavior that affect this still relatively unknown corner of the Web.

CCS Concepts

•**Security and privacy** → **Web application security;** *Domain-specific security and privacy architectures;*

Keywords

Dark web; Web security; Honeypot

1. INTRODUCTION

Based on the accessibility of its pages, the Web can be divided in three parts: the *Surface* Web – which covers everything that can be located through a search engine; the *Deep* Web – which contains the pages that are not reached by search engine crawlers (for example because they require a registration); and the more recent *Dark* Web – which is dedicated to websites that are operated over a different infrastructure to guarantee their anonymity, and that often require specific software to be accessed.

SAC 2017, April 03 - 07, 2017, Marrakech, Morocco

ISBN 978-1-4503-4486-9/17/04…$15.00

DOI: http://dx.doi.org/10.1145/3019612.3019796

The most famous "neighborhood" of the Dark Web is operated over the Tor network, whose protocols guarantee anonymity and privacy of both peers in a communication, making users and operators of (hidden) services in the Dark Web more resilient to identification and monitoring.

As such, over the last years, miscreants and dealers in general have started to adopt the Dark Web as a valid platform to conduct their activities, including trading of illegal goods in marketplaces, money laundering, and assassination [12]. Moreover, Tor has been reported to be leveraged in hosting malware [18], and operating resilient botnets [9]. While, to a certain extent, these studies have shown how the Dark Web is used to conduct such activities, it is still unclear if and how miscreants are explicitly conducting attacks against hidden services, like a web application running within the Tor network. While web attacks, or attacks against exposed services on the Internet are common knowledge and have been largely studied by the research community [10, 11, 22], no previous work has been conducted to investigate the volume and nature of attacks in the Dark Web.

To this extend, in this work we discuss the deployment of a high-interaction honeypot within the Dark Web to collect evidence of attacks against its services. In particular, we focus our study on web applications to try to identify how attackers exploit them (without a search engine for localization) and what their purpose is after a service has been compromised. Our preliminary measurement casts some light on the attackers' behavior and shows some interesting phenomena, including the fact that the vast majority of incoming attacks are unintentional (in the sense that they were not targeted against Dark Web services) *scattered attacks* performed by automated scripts that reach the application from the Surface Web through Tor2web proxies.

2. HONEYPOT IN THE DARK WEB

In this section, we discuss the main differences, in terms of advantages and disadvantages, between deploying and maintaining a honeypot in the Surface Web versus operating a similar infrastructure in the Tor network.

In fact, the anonymity provided by Tor introduces a number of important differences. Some are positives, and make the infrastructure easier to maintain for researchers. Some are instead negative, and introduce new challenges in the honeypot setup and in the analysis of the collected data.

Table 1 summarizes the five main differences between the two environments, mentioning their impact (on the deployment, operation, or on the results collected by the honeypot) and which environment (Dark or Surface Web) provides bet-

Area	Impact	Better in
Attack Identification	Results	Surface Web
Service Advertisement	Operation	Surface Web
Stealthiness	Deployment	Dark Web
Operational Costs	Deployment	Dark Web
Collected Data	Operation	Surface Web

Table 1: Advantages and Disadvantages of Operating a High-Interaction honeypot in the Dark Web

ter advantages in each category.

Attack identification

The most significant difference between a deployment on the surface Web and on the Tor network is the anonymity of the incoming requests. In a traditional honeypot, individual requests are typically grouped together in *attack sessions* [10] to provide an enriched view on the number and nature of each attack. A single session can span several minutes and include hundreds of different requests (e.g., to probe the application, exploit a vulnerability, and install post-exploitation scripts).

Since many malicious tools do not honor server-side cookies, this clustering phase is often performed by combining two pieces of information: the timestamp of each request, and its source IP address. Thus, requests coming in the same empirically-defined time window and from the same host are normally grouped in a single session.

Unfortunately, the source of each connection is hidden in the Tor network, and therefore the identification of individual attacks becomes much harder in the Dark Web. Moreover, if the attacker uses the Tor browser, also the HTTP headers would be identical between different attackers.

Stealthiness

If on the one hand the anonymity provided by the Tor network complicates the analysis of the attacks, on the other it also simplifies the setup of the honeypot infrastructure. In fact, a core aspect of any honeypot is its ability to remain *hidden* as the quality of the collected data decreases if attackers can easily identify that the target machine is likely a trap.

For instance, the nature of the Surface Web reveals information like the `Whois` and `SOA` records associated with a domain name, or the geo-location of the IP address the honeypot resolves to. To mitigate this risk, Canali et al. [10] employed a distributed architecture including hundreds of transparent proxy-servers that redirected the incoming traffic via VPN to the honeypots hosted on the researchers' lab. This solution successfully solve the problem of hiding the real location of the web applications, but it is difficult to maintain and requires the proxies to be located on many different networks (often on online providers).

Luckily, this problem does not exist on the Dark Web. The honeypot can run anywhere, without additional expedients as the Tor network is sufficient to guarantee the anonymity of the endpoints. Moreover, if a particular domain is blacklisted by the attackers, it is sufficient to generate a new private key/hostname pair to host content under a new domain name.

Service advertisement

As the most important value of a honeypot is the collected data, it is essential to attract a large number of attackers. On the Surface Web, it is typically the role of search engines to make the honeypot pages visible to the attackers interested in a certain type of target. For instance, honeypots often employ vulnerable versions of popular CMSs, as attackers routinely look for them by using Google Dorks [25].

It is also possible for a website on the Surface Web to attract attackers by simply placing some keywords or specific files as John et al. described in their work [14]. For example, including a known web shell or disclosing the vulnerable version of an installed application along with its name is a widely used strategy to lure attackers.

These popular "advertisement" approaches are not straightforward to apply to services hosted on the Tor network. As we later discuss in Section 4, it is still possible for `.onion` web sites to be indexed by Google. However, in order to gain popularity and attract attackers, researchers should carefully employ alternative techniques – such as advertising the website in forums, channels, or link directories specific to the Dark Web.

Operational costs

Since, as explained above, operating a honeypot in the Dark Web does not require any special domain registration or dedicated hosting provider, the total cost of the operation is typically very low. Canali et al [10] had to register hundreds of domain names (and routinely change them to avoid blacklisting) as well as several dedicated hosting providers – which are often difficult to handle because they often block the accounts if they receive complains about possibly malicious traffic.

In comparison, an equivalent infrastructure on the Dark Web only requires the physical machines where the honeypot is installed, as creating new domains is free and can be performed arbitrarily by the honeypot administrator.

Nature of the collected data

Some criminals use the Tor network to host illicit content like child pornography, since it protects both the visitors and the host by concealing their identities. Therefore, as we explain in Section 3, we had to take some special precautions to prevent attackers from using our honeypot to store and distribute this material. Unless researchers work in collaboration with law enforcement, these measures are required to safely operate a honeypot in the Dark Web.

3. HONEYPOT SETUP AND DEPLOYMENT

In this section we describe the setup of our honeypot. Our deployment is composed of three types of web-based honeypots and a system-based honeypot. Each of them is installed in a separate virtual machine (VM) hosted on our premises. The use of virtual machines allow us to revert the honeypots to a clean state after they are compromised. All honeypots are connected to the Tor network and made available as hidden services.

Note that each VM was fully patched to prevent privilege escalation, i.e. an attacker who compromised any of our machines would not be able to modify any system file and could

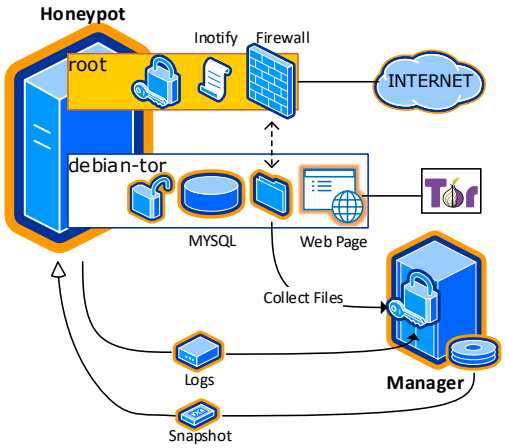

Figure 1: Simplified Honeypot Infrastructure

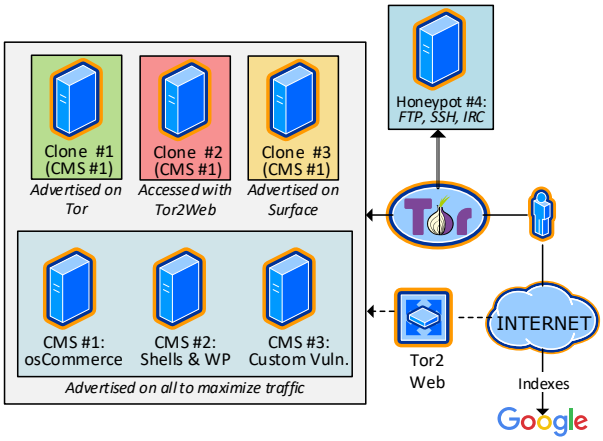

Figure 2: Initial deployment of honeypots with different advertisement strategies

only interact with the content of few selected directories[1].

Moreover, we used a set of firewall rules to restrict the attackers' network capabilities. In particular, we blocked all incoming and outgoing connections from all ports, except the ones used by Tor to operate, and ports associated to services that we explicitly offered. The firewall was also configured to enforce strict rate-limits to prevent denial-of-service attacks.

Web Applications

To mimic the setup used by a casual user, we decided to install all the applications in their default configuration, e.g. with all files located under `/var/www` and owned by the user `debian-tor`.

In each honeypot we installed `ModSecurity` [5], a popular monitoring and logging tool for the Apache web server. We configured ModSecurity to log the content of all HTTP POST requests along with their headers.

We also used a real-time file system event monitoring framework called `inotify` to detect all newly created/modified files, and copy them in a private directory for later inspection. Most importantly, using `inotify` we promptly detected, deleted, and shred any multimedia file uploaded by an attacker – to prevent our servers from hosting illegal material.

After we completed the configuration of our honeypots, we took a VM snapshot of their clean state. Later, every night, our system was configured to automatically retrieve all the files collected by `inotify` and a copy of all log files, and then to revert each VM to its original snapshot. A simplified representation of our honeypot infrastructure is given in Figure 1.

In order to bait the attackers, we decided to deploy three different honeypot templates:

1. **A website disguised as an exclusive drug marketplace that only trades between a close circle of invited members** – The website was realized using an old version of the popular OSCommerce e-marketing application. The version used in our experiments contains several known vulnerabilities, which allow an attacker to take over the admin panel and arbitrarily manipulate accounts and files.

2. **A blog site that advertises customized Internet solutions for hosting in the Tor network** – The website was realized using an old version of WordPress, which contained several known vulnerabilities.

   The honeypot also contained a number of sub-directories with different web shells, in order to mimic the fact that the site was already compromised by other attackers. The website was misconfigured to allow directory listing, so that an attacker (or an automated script) could easily navigate through the structure of the website and locate the shells.

3. **A custom private forum that only allowed registered members to login** – The website described the procedure to become a member, which required a valid reference from another existing member. In this case, we manually included a custom remote file-inclusion vulnerability that allowed an attacker to upload arbitrary files by manipulating PHP filters. The vulnerability was designed to be quite "standard", mimicking many existing vulnerabilities of the same type reported in other applications. However, it was also designed not to be straightforward to identify for automated scanners, as the goal of this third honeypot service was to collect information about manual attacks.

Although slightly tailored for our scenario in terms of advertised content, the first two templates were also used by a previous study of web honeypot [10]. These were intentionally chosen to be able to compare the types of attacks received on the Dark Web with those normally observed in the Surface Web.

The third template was instead specifically designed to avoid automated scanners and study more sophisticated attackers who may be interested in manually exploiting services hosted in the Dark Web.

We started by advertising our honeypot applications in three different ways: (i) by posting their URLs in several

[1]Excluding attacks leveraging 0-days and undisclosed vulnerabilities

Tor network's forums, channels, search engines and yellow pages, (ii) by visiting (twice a day) the applications via the Tor2Web proxy – which shares the visited URLs with Ahmia [2], a search engine for Tor, and (iii) by posting their URLs to several pages on the Surface Web.

In particular, to measure the success of our approaches, we deployed the first template three times (i.e., one for each advertisement technique). On top of that, we also deployed a copy of all templates by using a more aggressive strategy that includes all the three mechanisms described above.

Other Services

We also decided to include in our system a machine dedicated to collect system-level attacks directed towards other type of services appropriately configured to facilitate reconnaissances from attackers (e.g., by leaking the list of users via finger) or to expose weaknesses or misconfiguration.

This machine, reachable only over the Tor network, ran the following services:

1. We used `finger` to broadcast the list of active users and we provided a file containing the hashed version of a user's password on an open FTP server. We also used message-of-the-day informative to advertise our honeypot as a file-server.

2. We offered an open (anonymous) FTP server. We served a valid upload directory (`incoming`) for hypothetical illicit uses like drop-zone and exploitation, and we provide some documents for download, one of which contained the password for one of the system users.

3. We enabled SSH login on 2 users. The shell was chroot jail protected. Both accounts were easily guessable, i.e. the first having a straightforward name and password combination (`guest:guest`); the second having the base64-encoded version of the password leaked in the FTP document.

4. IRC. Chats are known to be used as rendez-vous points to discuss illicit offers (e.g. stolen accounts) or host C&C servers of botnets. With the intent of understanding whether attackers would try to abuse chats in the Tor network, we installed an open IRC service (`UnrealIRCd`) and registered anonymous logins.

This machine was also advertised using all the previously described channels (for the Tor2web case, we used the proxy to access a static webpage hosted on the honeypot, describing the machine as a Dark Web file hosting server). Figure 2 shows a summary of the initial deployment strategy used in our experiments.

4. DATA COLLECTION AND ANALYSIS

We run our experiments over a period of seven months between February and September 2016. Due to maintenance and re-configuration of the honeypots, the individual honeypot services were online for a total of 205 days.

The experiments were divided in three phases. During the first phase (which lasted for 37 days until the end of March) we applied the three advertisement strategies described in Section 3 on a single honeypot template (CMS #1), to measure their impact on the incoming traffic and on the number of attacks. In the second phase (from the 1st of April to end of May) we advertised the three templates using all available strategies, to maximize the amount of collected data. Finally, for the last four months of experiments, we restricted the access to our honeypot by blocking Tor2web proxies, in order to exclusively focus on attacks within the Tor network.

	Clone #1 (Tor Only)	**Clone #2** (Tor2Web)	**Clone #3** (Surface Only)
GET	3.29M	1.26M	1.02M
POST	20	147	1

Table 2: Number of GET & POST requests for different advertisement strategies

In the rest of the section we describe the impact of these three factors on the collected data: the advertisement strategy, the source of the attack (from the Surface or the Dark Web), and the type of honeypot template.

4.1 Impact of Advertisement Strategies

As we mentioned in Section 3, we created three clones of our first honeypot template (CMS #1), which we then advertised using different channels.

In Table 2, we present the total number of requests received by the three clones. Quite interestingly, all clones received a comparable amount of overall traffic (between 1 and 3.3M hits). However, looking at the POST requests the picture is quite different. For instance, the honeypot advertised on Tor only received over 3M GET requests but only 20 POSTs. The first number is inflated because the same visitor may have requested multiple resources – and we already discussed how difficult it is to track visitors in the Dark Web, when using the same browser and no endpoint information are available. In addition, since attackers required a POST request eventually to upload their files, we decided that looking at POST requests was a better way to estimate the "interesting" traffic, and filter out most of the harmless visitors, automated Internet scanners, and other forms of background noise.

Finally, it is interesting to note how the second clone – advertised through Tor2web, was the only one to receive attacks (over 20) in this first phase of our experiments.

4.2 Role of Tor Proxies

The Tor2web[2] projects provide a simple way for users to access resources on the Dark Web by simply appending special extensions to onion domains. These special domains (such as `.onion.to`, `.onion.link`, and `.onion.city`) resolve to one of the Tor2web operators which in turn act as proxies from the Surface Web to the Dark Web. These services facilitate the access to the Tor network with the disadvantage of sacrificing the anonymity of their users.

Since Tor proxies make hidden services in Tor reachable with a normal HTTP request over the Internet and with *no* additional configuration, they can be used by traditional browsers but also by automated scripts and crawlers. The presence of these proxies turned out to be extremely important for our experiments. In fact, once a proxy domain is indexed by a search engine, the target website can be located using traditional Google Dorks [23] and therefore becomes implicitly a target of automated exploitation scripts [10].

[2]https://www.tor2web.org/

Proxy	Online	Transparent
*.onion.to	✓	✗
*.onion.link	✓	✗
*.onion.city	✓	✗
*.onion.nu	✓	✓
*.onion.cab	✓	✓
*.onion.direct	✗[1]	*Unknown*
*.onion.lt	✗[2]	*Unknown*
*.onion.sh	✗	*Unknown*
*.onion.ink	✗	*Unknown*
*.tor2web.org	✗	✗
*.tor2web.fi	✗[3]	✗
*.onion.rip	✓	✗

[1] discontinued
[2] website is offline
[3] redirects to tor2web.org

Table 3: List of inspected Tor proxies

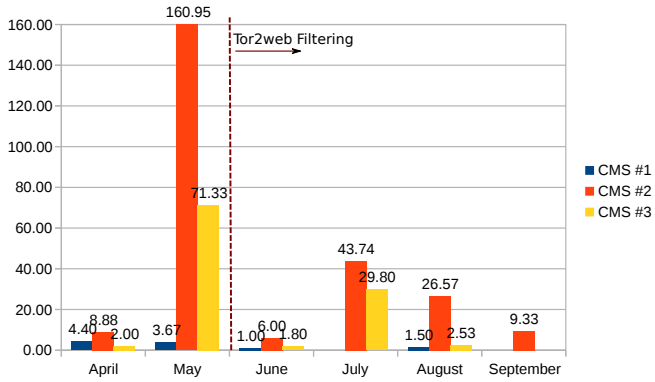

Figure 3: Average number of daily post requests

Once we noticed this phenomenon and the fact that the vast majority of the attacks indeed came through these proxy services, we decided to block the request coming from Tor2web.

Table 3 shows a list of different operators, mentioning those that were online during our experiments and those that we could identify by looking at the HTTP headers they append. For example, some of the Tor2web operators like `.onion.to`, `.onion.link`, and `.onion.city` includes extra headers in the request field (such as `HTTP_X_TOR2WEB`, `HTTP_X_FORWARDED_PROTO` and `HTTP_X_FORWARDED_HOST`). This allowed us to identify and block the requests coming from these services, by serving them a static page explaining that our services were only available from the Tor network.

This change – implemented from June in our experiments, lead to a sharp drop in the number of incoming requests and in the number of incoming attacks (see Figure 3).

Interestingly, blocking Tor2web proxies also had a clear effect on the *type* of files uploaded by the attackers. For example, we stopped receiving phishing kits or mailers (see Section 5 for more details) after we implemented our blocking strategy. Therefore, it is safe to say that even though it was possible that some requests still came through transparent proxies, our countermeasure was able to effectively prevent most of the automated attacks specific to the Surface Web– which, indeed, was our initial goal.

	CMS #1 (OsCommerce)	CMS #2 (Shells & WordPress)	CMS #3 (Custom Vuln.)
Tor2web	115 (8 days)	1,930 (23 days)	0
TOR	0	2,146 (79 days)	689 (5 days)

Table 4: Number of attack-related POST requests

4.3 Honeypot Templates

As we explained in Section 3, we used three honeypot templates based on different web applications and, more importantly, with different types of vulnerabilities.

Table 4 shows the number of attack-related POST requests along with the number of days in which we received at least a single attack. We consider a request as attack-related if it contains an attempt to exploit a vulnerability or if it is involved in the post-exploitation phase – for example, to upload further files on the exploited application or to inspect the host through a webshell.

Predictably, the web shells installed on CMS #2 served as bait to attract the highest number of attack-related requests. CMS #1, instead, received less attacks and all originating through proxy services likely as the result of using Google Dorks.

Interestingly, some attackers used dorks on Tor-specific search engines. For instance, an attacker found one of the honeypots through a Tor search engine[3] by simply querying the set of keywords *"Index of /files/images/"*.

Finally, CMS #3 (i.e., the one containing our custom vulnerability) received the lowest number of attacks and none of them actually succeeded. Even more interestingly, while CMS #1 was only attacked from the Surface, CMS #3 was only attacked from the Tor network. This is in line with our expectations: well-known vulnerable CMSs are targeted by automated scripts using dorks, while custom websites are only targeted by humans or dedicated scanners (which in our case were run in the Tor network).

Afterward, we manually analyzed all POST requests to identify the successful attacks, i.e. those in which the attacker successfully compromised the honeypot application. CMS #1 was successfully compromised by attacks originating from the Surface Web. Out of 115 attack-related requests, 105 (91%) of them were successful attacks. After we started blocking Tor proxies, we did not observe further attacks.

For CMS #2, we counted an attack as successful if it exploited the WordPress vulnerability or if it used one of the existing webshell to install or modify a file in the system (i.e., a simple inspection of the system was not considered a successful attack). Overall, 1,255 (65.0%) of the attack-related requests originating through Tor2web succeeded. On the contrary, only 154 (7.2%) of the ones coming through the TOR network succeeded. This is a very interesting result, and it shows that the majority of the attackers who interacted with our shells from the Surface Web ended up performing some change on the system. The attackers from the Tor network instead mostly inspected the system and moved away without touching any file (more about this phenomenon will be discussed in Section 5).

[3] http://hss3uro2hsxfogfq.onion

CMS #3 never received a single attack from Tor2web, if we exclude some manual attempts to guess a valid username and password (which we did not count as an attack in our statistics). As we mentioned in the beginning of this section, this one never received a successful attack.

5. ATTACK EXAMPLES

Over the entire experiment, attackers uploaded on our honeypot 287 files (an average of 1.4 per day). In comparison, Canali et al. [10] collected over 850 files per day – but they used 500 clones (against the 3 we used in our experiments). In fact, the goal of our study was not to collect a large number of attacks, but rather to study their nature and how the effect of different factors like the advertisement channel, the type of service, and the source of the attack affected.

In the following sections, we classify the attacks into three different categories: automated scattered attacks from the Surface Web, automated attacks from the Tor network, and manual attacks. For each category, we present some examples and we discuss in more details the attacks most commonly observed in our honeypots.

5.1 Scattered attacks

As we described previously, regular search engines unexpectedly index web pages hosted on the Dark Web through Tor2web proxies. As a result, websites located in the Tor network receive part of the background noise of automated attacks that plague the Web, scattered through the proxies that act as gateways between the two "sides" of the Web.

For this reason, the vast majority of the attacks observed in our honeypot were simple, and very similar to what was observed by previous studies [10, 14]. Basically, the modus operandi of these attackers consisted of locating our websites using Google Dorks, and employing automated scripts to visit the pages, exploiting the known vulnerabilities, and possibly uploading files for the next phase of the attack. In the majority of the cases, these attacks involved the use of web shells, which allowed the attackers to later run system commands on our honeypot. Using these web shells, attackers could upload other files including web mailers, defacement pages, and phishing kits.

The completely automated nature of these attacks and the types of files uploaded in the honeypot make us believe that in the majority of the cases the attackers were not even aware of the fact that they compromised applications hosted in the Dark Web.

Web shells – We collected 157 unique variations of web shells uploaded by the attackers. This was an expected behavior since most of the time the attackers made use of automated scripts for the first phase of the attack. We also observed that once a web shell was deployed, other shells were often uploaded using this first web shell, over a short period of time. Usually, while the initial shell was unobfuscated, the subsequent ones were protected with a password. Some of the collected web shells were base64-encoded and they were configured to de-obfuscate at run time by means of the PHP's *eval* function.

Phishing kit & Mailers – Surprisingly, attackers uploaded six phishing kits for popular targets (in particular Paypal). Having a phishing kit for such applications does not make much sense in our setting, since there is no Paypal on Tor to begin with. But the fact that all the phishing-related attacks were coming through TOR proxies, strengthen our hypothesis that the attackers (or their automated scripts) were probably not aware of the location of the exploited application. Similarly, 22 mailers were uploaded through Tor proxies, but none of them was ever used by the attackers.

Defacements – Our web applications got defaced 33 times. Usually, a web shell was uploaded before the defacement and subsequently the index page was modified or a brand new one was uploaded by using the prior web shell. From an analysis, this process looked automated since the same pattern was observed multiple times with the same defacement page. Half of the attacks originated from the Surface Web, and the rest came directly from the Tor network.

In one defacement specific to Dark Web, the defacer modified the index page of CMS#2 to promote one of his sites called Infamous Security[4], where the authors apparently advertise their hacking services.

5.2 Automated Attacks through Tor

Automated Scans – Our honeypots received over 1,500 path traversal attempts (e.g. to fetch `../../etc/passwd`, or `../../etc/vmware/hostd/vmInventory.xml`). As we could infer from the User Agent, attackers seemed to be using the NMap[5] scripting engine for scanning their targets.

Access to the Service Private Keys – One of the most common scan attempt we received within the Tor network was the download of the *private key* that we voluntarily hosted on the web applications' root directory.

Every time the Tor service starts, it creates a private key (if not existing) and assigns the corresponding hidden-service descriptor (i.e. the hostname) to this private key. While the private key must *not* be accessible with default Tor and Apache configurations, in our case we intentionally misconfigured the service to let it accessible from the Web for CMS #3 starting from mid Agust. Exposing a private key simply means that the owner risks losing the hostname to the adversary (and therefore potentially all incoming traffic). Thus, the first and the easiest automated attack is to fetch this key, if its location and permission are not configured correctly.

During the operation of our honeypot, we observed and confirmed over 400 attemps to fetch the private key. Attackers could use those keys to impersonate our honeypot and conduct attacks like phishing or hosting of malware.

Other Services – We reported a number of successful connections to our FTP, SSH, and IRC services that, most likely, represent instances of banner grabbing or information gatherings. In total, we confirmed 74 SSH connections (client-side terminated or timeout), 61 successful FTP (anonymous) logins and 91 IRC logins.

5.3 Manual attacks

Post-Exploitation Actions – We noticed that attackers connecting via Tor network (instead of using Tor proxies) were generally more careful and spent more time to investigate the environment. For instance, their first action when using a web shell was to gather additional knowledge by listing directories, checking the content of the local database,

[4]http://5eaumbq2k6yc4sjx.onion/

[5]http://nmap.org

fetching `phpinfo` and system files such as `crontab`, `passwd`, `fstab` and `pam.conf`.

Such attackers never went beyond exploring the system, compared to the ones we mentioned in Section 5.1 – who almost always installed additional components. In fact, manual attackers from the Tor network often deleted their files and left the honeypot after their initial inspection. In few cases, the attackers also left messages (such as "Welcome to the honeypot!") or redirected our index page to a pornographic video. In one example, the attacker downloaded 1GB of random data from a popular website to test the network download speed and renamed the file as 'childporn.zip' – supporting the fact that many attacks from Tor were manually operated and resulted in a successful identification of the honeypot.

While these cases support the fact that there are people manually exploiting websites on the Dark Web, all these attacks used previously installed web shells or extremely popular CMS vulnerabilities. None of them was able to exploit the (still relatively simple) custom vulnerability on CMS #3.

FTP and SSH – Overall, we identified 71 FTP file downloads. Interestingly, all occurred in a sub-directory and *none* on the root directory of the server – showing the manual nature of the action and the interest in accessing specific data. In one case, the miscreant used our bait login credentials included in our honey-document to log in to the SSH server. This was an interesting scenario, in which the attacker was able to manually extract information collected from one service to connect to another service.

Even more interestingly, the attacker first connected to the SSH server sending his real username for the login, likely due to the fact that this is performed automatically by ssh clients. The attacker then immediately killed the session and reconnected with the correct username previously gathered from the honey-document.

Attacks Against the Custom Application – The application with the custom vulnerability received little attention through the entire experiment. Except for some automated background noise of SQL injections and directory traversal attempts, we noticed 87 requests (GET and POST) that attempted to tamper with the parameter vulnerable to remote file inclusion, but without any success. One attacker analyzed the entire website using the Acunetix [1] web vulnerability scanner but the tool was unable to exploit the vulnerability. Another attacker focused on the login form and run the sqlmap[6] tool to try to detect a possible SQL injection, again without success.

6. RELATED WORK

Attacks against web applications have already been studied by using either low-interaction [3, 6, 14, 20] or high-interaction web honeypots [4, 10, 11, 22]. However, all these works targeted the Surface Web and we believe we are the firsts to document attacks in the Dark Web by mean of the practical deployment of a high-interaction honeypot.

A related set of studies focused on measuring the characteristics of the Dark Web, including its size, the connection between websites, and the services (i.e. protocols) provided over the Tor network. OnionScan, for example [16], leverages hyperlinks contained in web pages and other features (like correlation on ssh fingerprints and ftp banners), to build relationships among hidden services. This dataset consists of about 5,600 active sites that were scanned in June 2016.

In a similar work, Ciancaglini et al. [12] actively crawled the Dark Web for a period of two years and reported on the cyber-criminal services and illicit goods available in the Dark Web like marketplaces, laundering services for cryptocurrencies, and hosting platforms for malware.

When it comes to attacks in the Dark Web, or against the darknets used to operate the Dark Web, a consistent amount of literature has been produced. A first class of papers propose attacks aimed at de-anonymize hidden services, e.g. by recovering the public IP address on which the hidden service operates. CARONE [17] makes use of heuristics to match information like keywords in the content of the hidden service and certificates chain with candidate Internet endpoints, then validated in a second phase.

Kwon et al. [15] propose an attack in which a combination of website fingerprinting and circuit fingerprinting techniques are used to de-anonymize hidden services. While website fingerprinting is already widely used (e.g., in the Surface Web), authors revealed that during the circuit construction phase between clients and hidden services, darknets as Tor exhibit fingerprintable traffic patterns that allow an adversary to efficiently and accurately identify and correlate circuits involved in the communication.

Panchenko et al. [19] propose a more general approach that identifies the content of encrypted and anonymized connections (e.g., Tor) by observing patterns of data flows such as packet size and direction. Other researchers recently hit the media when they revealed their ability in de-anonymize users and hidden services in Tor [7, 8].

A different class of attacks has been analyzed in [24] and [21]. Winter at al. [24] document malicious exit relays in the Tor. The authors developed *exit relays scanners* for credential harvesting and MitM attacks, and used them to identify malicious exit relays nodes. More recently, Sanatinia et al. [21] exposed another category of misbehaving Tor relays (HSDirs) that are integral to the functioning of the hidden services.

On top the Dark Web-specific attacks described so far, denial of service (DoS) against hidden services has been reported in the wild [13]. In fact, with the increase on the number of business-related websites been deployed in hidden services, well-understood attacks (like DoS) are seen occurring in the Dark Web. What is not clear, so far, is how much a hidden service is exposed to threats like web-based attacks (e.g. SQLi, path traversal, etc..), bruteforce attacks, and how these attacks are conducted in the Dark Web– e.g., if manually or automatically.

7. CONCLUSIONS

This paper discusses the deployment of a high-interaction honeypot in the Tor network, to explore the modus operandi of attackers in the Dark Web. We conducted our experiments in three different phases over a period of seven months and we assessed the effectiveness of advertisement strategies on the number and nature of the attacks. Our preliminary results show that also hidden services can receive automated attacks from the Surface Web with the help of Tor proxies. Moreover, we found that miscreants in the Dark Web tend to involve more manual activity, rather than relying only on automated bots as we initially expected. We hope that our

[6] http://sqlmap.org

work will raise awareness in the community of operators of hidden services.

8. REFERENCES

[1] Acunetix Ltd, Web Vulnerability Scanner. http://www.acunetix.com/vulnerability-scanner/. Accessed: 2016-09-26.

[2] Ahmia. https://ahmia.fi/. Accessed: 2016-09-26.

[3] Google Hack Honeypot. http://ghh.sourceforge.net/. Accessed: 2016-09-26.

[4] High interaction honeypot analysis tool. https://sourceforge.net/projects/hihat/. Accessed: 2016-09-26.

[5] ModSecurity: Open Source Web Application Firewall. https://www.modsecurity.org/. Accessed: 2016-09-26.

[6] MushMush Foundation. http://mushmush.org/. Accessed: 2016-09-26.

[7] Tor Project. Did the FBI Pay a University to Attack Tor Users? https://blog.torproject.org/blog/did-fbi-pay-university-attack-tor-users.

[8] B. H. U. 2014. You Don't Have to be the NSA to Break Tor: Deanonymizing Users on a Budget.

[9] D. Brown. Resilient botnet command and control with tor. 2010.

[10] D. Canali and D. Balzarotti. Behind the scenes of online attacks: an analysis of exploitation behaviors on the web. In *Proceedings of NDSS 2013*, pages n–a, 2013.

[11] O. Catakoglu, M. Balduzzi, and D. Balzarotti. Automatic extraction of indicators of compromise for web applications. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, 2016.

[12] V. Ciancaglini, M. Balduzzi, R. McArdle, and M. Rösler. Below the Surface: Exploring the Deep Web [Technical Report]. http://www.deepweb-sites.com/wp-content/uploads/2015/11/Below-the-Surface-Exploring-the-Deep-Web.pdf.

[13] T. Fox-Brewster. Tor Hidden Services And Drug Markets Are Under Attack, But Help Is On The Way. http://www.forbes.com/sites/thomasbrewster/2015/04/01/tor-hidden-services-under-dos-attack/.

[14] J. P. John, F. Yu, Y. Xie, A. Krishnamurthy, and M. Abadi. Heat-seeking honeypots: design and experience. In *Proceedings of the 20th international conference on World wide web*. ACM, 2011.

[15] A. Kwon, M. AlSabah, D. Lazar, M. Dacier, and S. Devadas. Circuit fingerprinting attacks: Passive deanonymization of tor hidden services. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 287–302, 2015.

[16] S. J. Lewis. OnionScan Report June 2016 - Snapshots of the Dark Web. https://mascherari.press/onionscan-report-june-2016/.

[17] S. Matic, P. Kotzias, and J. Caballero. Caronte: Detecting location leaks for deanonymizing tor hidden services. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15. ACM, 2015.

[18] P. H. O'Neill. Bank thieves are using Tor to hide their malware [News]. http://www.dailydot.com/crime/bank-malware-tor2web/. Accessed: 2016-09-26.

[19] A. Panchenko, F. Lanze, A. Zinnen, M. Henze, J. Pennekamp, K. Wehrle, and T. Engel. Website fingerprinting at internet scale. In *Proceedings of NDSS 2016*, 2016.

[20] N. Provos. A virtual honeypot framework. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 1–1, Berkeley, CA, USA, 2004. USENIX Association.

[21] A. Sanatinia and G. Noubir. Honions: Towards detection and identification of misbehaving tor hsdirs. https://www.securityweek2016.tu-darmstadt.de/fileadmin/user_upload/Group_securityweek2016/pets2016/10_honions-sanatinia.pdf.

[22] O. Starov, J. Dahse, S. S. Ahmad, T. Holz, and N. Nikiforakis. No honor among thieves: A large-scale analysis of malicious web shells. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, 2016.

[23] F. Toffalini, M. Abba, D. Carra, and D. Balzarotti. Google Dorks: Analysis, Creation, and new Defenses. July 2016.

[24] P. Winter, R. Köwer, M. Mulazzani, M. Huber, S. Schrittwieser, S. Lindskog, and E. Weippl. Spoiled onions: Exposing malicious tor exit relays. In *International Symposium on Privacy Enhancing Technologies Symposium*. Springer, 2014.

[25] J. Zhang, J. Notani, and G. Gu. Characterizing google hacking: A first large-scale quantitative study. In *International Conference on Security and Privacy in Communication Systems*. Springer, 2014.