Monkey-Spider: Detecting Malicious Websites with Low-Interaction Honeyclients

Ali Ikinci Thorsten Holz Felix Freiling

University of Mannheim
Mannheim, Germany

ali.ikinci@contentkeeper.com
{holz|freiling}@informatik.uni-mannheim.de

Abstract: Client-side attacks are on the rise: malicious websites that exploit vulnerabilities in the visitor's browser are posing a serious threat to client security, compromising innocent users who visit these sites without having a patched web browser. Currently, there is neither a freely available comprehensive database of threats on the Web nor sufficient freely available tools to build such a database. In this work, we introduce the *Monkey-Spider* project [Iki]. Utilizing it as a client honeypot, we portray the challenge in such an approach and evaluate our system as a high-speed, Internet-scale analysis tool to build a database of threats found in the wild. Furthermore, we evaluate the system by analyzing different crawls performed during a period of three months and present the lessons learned.

1 Introduction

The Internet is growing and evolving every day. More and more people are becoming part of the so-called *Internet community*. With this growth, also the amount of threats for these people is increasing. Online criminals who want to destroy, cheat, con others, or steal goods are evolving rapidly [Ver03]. Currently, there is no comprehensive and free database to study malicious websites found on the Internet. Malicious websites are websites which have any kind of content that could be a threat for the security of the clients requesting these sites. For example, a malicious website could exploit a vulnerability in the visitor's web browser and use this to compromise the system and install malware on it.

HoneyMonkey [WBJ+06] and *SiteAdvisor* [McA] are proprietary systems with an approach to build such research databases. Nevertheless, they are not freely available for further research and only limited access to their results is publicly provided. In this work, we introduce the *Monkey-Spider* project. The task which we want to address with this work is to build an easy-to-use infrastructure to detect and monitor malicious websites and to evaluate it on popular parts of the World Wide Web. The resulting system will be referenced further in this article as *Monkey-Spider*.

One of the best tools for real-world analysis of malicious activities and techniques are *honeypot*-based technologies. A honeypot is an information system resource whose value

lies in unauthorized or illicit use of that resource [Pro04]. Honeypots bait attackers to malicious actions and then examine their behavior and capture their tools. Honeypots can be distinguished as *server honeypots* and *client honeypots*: Server honeypots are computer systems serving different kinds of services, e.g., via the protocols HTTP or FTP, to clients. They wait for clients to attack them. On the other hand, client honeypots are clients requesting contents from other servers and thus initiate malicious servers to attack them. Starting from this idea, the Monkey-Spider system will be used in our work as a client honeypot (or *honeyclient*) for web servers, to find, observe, and capture malicious activities on the publicly accessible World Wide Web.

Manual analysis for individually detected malware sites have appeared in the last years [Fra04, FS05]. These analyses are expensive, time-consuming, and far too few. Our goal is the automated, large-scale analysis for millions of websites. To build an infrastructure for studying actual threats on the Internet, the resulting system should be able to (or at least help to) answer the following questions:

- How much malware is on the Web?
- Where is that malware located?
- How are the threats changing over time?
- What is the topic specific probability for an infected site?

This paper is outlined as follows: In Section 2, we provide an overview of related work. Section 3 introduces the Monkey-Spider project and describes its components in detail. We present results collected with the help of Monkey-Spider during a period of three months in Section 4 and conclude with an overview of future work in this area in Section 5.

2 Background and Related Work

Honeypots [Spi02] are dedicated deception devices. The value of these systems lies in being probed, attacked, and compromised by manual or automated attacks to gather information on the attacker without his knowledge. Being dedicated only for attack analysis and not for production use, every action on a honeypot is suspicious and illegal by definition.

In the area of honeypots, we can distinguish between *server* and *client honeypots*. *Server honeypots*, who are often just called honeypots, wait for the adversary to attack and thus have to be an interesting target for an attacker to fulfill its task. Like in the real world an "open door may tempt a saint", such a system should provide publicly known network service vulnerabilities and weak security profiles like weak or no passwords, old and unpatched system software etc. to succeed. On the other hand, *client honeypots*, which are also sometimes denominated as *honeyclients*, are the opposite to server honeypots. Honeyclients actively crawl or access the Web to search for servers that exploit the client and thus to gather information of how attackers exploit clients. Even though web browsers are

not the only kind of client software that can be attacked, the current focus of this work and other honeyclients is mostly based on the analysis of Web client exploitation.

One major drawback of server honeypots is that they have to wait for an attack: an attack appears only by chance, thus it is possible that a ready to attack honeypot is not attacked for months or it is quite possible that it is attacked occasionally by many attackers at the same time. It is not easily predictable how frequently attacks will occur on a honeypot and thus the analyses get more complicated. In comparison with this behavior, honeyclients initiate every analysis and thus control the maximum number of possible attacks. The analysis is important even if no attack occurs because such websites and servers can then be classified as likely not to be malicious and thus a *safe* Web could be mapped.

Honeypots are usually classified as either *low-interaction* or *high-interaction* honeypots: High-interaction honeypots are real systems, providing real applications for the hacker to interact with. They are used to gather profound information about threats and attacks. Low-interaction honeypots *emulate* real systems and services. An attacked system has only restricted interaction capabilities and thus lacks some of its deception values.

Honeyclients can further be classified as low-interaction or high-interaction honeyclients: high-interaction honeyclients are usually real automated Web browsers on real operating systems which interact with websites like real humans would do. They log as much data as possible during the attack and allow a fixed time period for an attack. Since high-interaction honeypots provide detailed information about attacks, they are usually very slow and not able to scan large portions of the Web. Low-interaction honeyclients, on the other hand, are usually emulated Web browsers – for example Web crawlers – which do have no or only limited abilities for attackers to interact with. Low-interaction honeyclients often make use of static signature or heuristics based malware and attack detection and thus may lack the detection of zero-day exploits and unimplemented attack types. These honeyclients are not suited for an in-depth investigation of the actions of an attackers after a successful compromise because the system is only simulated and any other action than the initial exploitation is likely to be missed. In spite of these drawbacks, low-interaction honeyclients are often easier to deploy and operate and are very performant. They can be used for automatic malware collection and to take a sounding on a portion of the Web. Furthermore, the containment of attacks is easier compared to high-interaction honeyclients because the compromised system is not real and thus unusable for the attacker, which additionally simplifies the deployment of a low-interaction honeyclient. In Table 5 (Appendix), we provide a detailed comparison of different honeyclient solutions. In addition, a study by Provos et al. presents results from a large-scale measurement on malicious websites available on the World Wide Web [PMM+07]. Based on Google's cache of crawled websites, the authors can analyze in detail malicious websites with the help of honeypots. In our approach, we collect and analyze all data with the Monkey-Spider system.

3 Monkey-Spider Project

In this section, we describe Monkey-Spider in detail and provide an overview of the different components from this project. Our system can be classified as a crawler-based, low-interaction honeyclient. The main idea of Monkey-Spider is to first crawl the contents of a website and then analyze the crawled content for maliciousness. We do not want to analyze websites while surfing them with real Web browsers. The reason for not doing this is to be able to split up both tasks (crawling a website and analyzing the content) and optimize them separately for speed. Hence the scanning and the crawling can be done on different machines and thus combined more performant. The scanning usually does take significantly more time and resources than the crawling. The crawling process can take as much processing time and memory as the Internet connection can bear and the tuning of the crawler allows.

3.1 System Architecture

The Monkey-Spider system utilizes many existing freely available software systems. We provide in this section a brief overview of the project before presenting more details on each building block. Figure 1 provides an overview of the whole system architecture. The architecture is divided in different functional blocks, marked as dotted or drawn through rounded rectangles. These blocks can either be located on different computer systems for scalability and performance reasons or used on one computer system for the ease of use. The hatched modules are still in development.

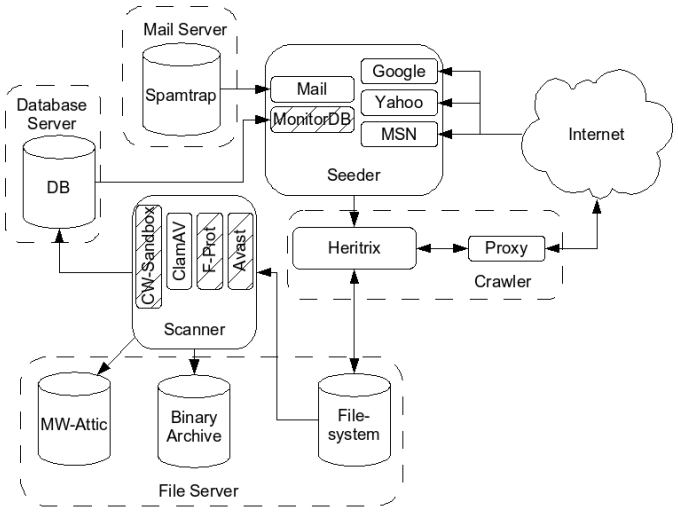

Figure 1: Schematic overview of the Monkey-Spider

Queue / Seed Generation. Every run begins with the *Seeder block* which generates starting URLs for the crawler, the so called *queue*. One can either generate seeds with the *Web search seeders* and/or generate seeds out of spam mails with the *mail seeder*. Another possibility is to use the *MonitorDB seeder* to re-queue previously detected malicious sites. The Web search seeders use the search engines Yahoo, Google and MSN Search for searching and URL extraction. The mail seeder extracts URLs out of collected spam mails from a spamtrap. In addition, the MonitorDB seeder is used to constantly re-seed previously found malicious content over time from our malware database.

Web Crawling. We use *Heritrix* [her], the Web crawler from the Internet Archive [IA], to crawl the URLs extracted by the Seeder and download content from the World Wide Web. Heritrix queues the generated URLs from the seeder and stores the crawled contents on the file server while generating detailed log files. Optionally we can use Heritrix through an interconnected Web proxy: the Web proxy can be used to increase performance and avoid duplicate crawling.

Malware Analysis. In the next step, we analyze the downloaded content. The actual data is stored by Heritrix in the so called ARC file format [ARC]. We extract the information from these files and then analyze the content with different anti-virus solutions and malware analysis tools. Identified malware and malicious websites are stored in the *malware attic* directory. In addition, information regarding the malicious content is stored in a database. Furthermore, we copy every found binary and JavaScript file to an additional archive directory for additional research purposes.

3.2 Seed Generation

In the step of seed generation, we try to find some starting points for our crawl. We use the following techniques for generating seeds.

Web Search Seeding. We use the Web Services APIs of the three popular search engines Google, Yahoo, and MSN Search. With the help of these interfaces, we retrieve URLs of the corresponding Web searches (typically the first 1000 hits). We chose some topics like “celebrity” or “games” as start points and search for typical terms within these topics. For example, to populate search results for “pirate” websites, we use the search terms ”crackz”, ”serialz”, ”warez” etc. After the search has finished, we collect all found URLs of one topic and use this list as seeds for our crawl. With this method, we can find popular pages for the given topic because Internet users typically either use a search engine as a starting point for surfing the Web or start from a previously known URL. They commonly use the first tens results of search results, thus our approach should cover typical starting URLs. By using a search engine, we have a certain guarantee that the found URLs are likely to have the right content for the topic we are searching for.

Spamtrap Seeding. Often malware propagates utilizing spam. A spamtrap is an email account established specially to collect spam mails. Such email addresses are planted openly on different places of the Internet, where spammers are likely to find them. Our idea is to collect spam mails in a spamtrap and then to extract all the URLs contained in the messages to use them as seeds for our crawl. Therefore we set up an email account intended to be a target for spammers. We then download the spam mails and extract found URLs in the messages. The extracted URLs are used as seeds for our crawl. Currently, we do not examine attached files, but this could be integrated as part of future work.

Blacklist Seeding. A general strategy to prohibit access to known ”bad” sites is to list them in a so called *blacklist*. This blacklist is then used as part of an access control mechanism, prohibiting access to all listed websites. There are several websites like [Mik] which generate blacklists for the Internet community, in order to protect themselves from malicious content and other unwanted parts of the Web. We have implemented a tool to automatically download blacklists from some major blacklists providers and use them as seeds for our crawls. It is obvious that these hosts have more malicious content than common websites. Thus these seeds do not represent the Web as a whole, but they represent the ”bad” guys and ”bad” company's who built a kind of business model around this practices. Blacklist files do generate a huge hit rate and deliver us many malicious contents for our research.

3.3 Web Crawling

Crawling is an important aspect because it is responsible for the scope and significance of our research. The *crawl scope* denotes how far we want to go in discovered URLs. Two parameters determine the crawl scope: the first one is the maximum link hops to include, which means not more URLs than this number of links from a seed will be included in a crawl. The second one is the maximum transitive hops to include, meaning URLs reached by more than this number of transitive hops will not be ruled in-scope, even if otherwise on an in-focus site. These two parameters are decisive settings for the hit count and the relevance. The hit count specifies how many malicious sites are found. The relevance is determined via the content analysis step.

After doing an evaluation of several web crawl solutions, we chose Heritrix. This tool is part of the Internet Archive's open source, extensible, Web scale, archival quality and easily customizable Web crawler project. For our development and evaluation we used the version 1.12.x of the stable release. One important feature of Heritrix is *link extraction*: to reach as many available content as possible, not only URLs contained in a HTML file are analyzed, but also any other content. Link extraction is the process of extracting URLs out of any document like HTML, JavaScript, PDF etc. The extraction step is essential for the scope of our evaluation because often related links are embedded in other files than the HTML file and are reachable for human surfers, too. Another important feature of Heritrix is *URL normalization*: syntactically different URLs can represent the same content on the Web. To avoid overhead caused by multiple downloads of the same resource, we have

to use several URL-normalization techniques. Heritrix has a dedicated module for this task and implements several well-known and widely used URL-normalization techniques, e.g., to strip known session IDs or any 'userinfo' found. We extended Heritrix with a web interface to have an administrator interface for Monkey-Spider. Furthermore, we implemented our own tools to handle the data collected via Heritrix. A comparison of the run time of the tool shipped with Heritrix and our own version revealed an acceleration of the extraction process with a factor of 50 to 70 times.

3.4 Content Analysis

With our approach of high-bandwidth and broad scope crawling, we have observed our crawler to crawl about 30 URLs/sec. With such high performance, we need fast and reliable tools for automatic malware analysis because the analysis seems to be the most time consuming part in our processing chain. Depending on the status of existing malware, our analysis can be separated into two states. The first approach is the detection of known malware with common antivirus scanners like ClamAV or others. In the second approach we analyze suspicious binaries with automatic, behavior-based malware analysis tools, like the CWSandbox [WHF07]. Furthermore, the Monkey-Spider system can be extended with any other automated analysis tool. The information regarding every detected malware is comitted to our database running MySQL, which itself does not have to have any highly optimization due to the small amount of found malware up to now. But this could be necessary if a couple of malware scanning machines try to commit their results to one centralized database.

3.5 Limitations

The World Wide Web is a system of interlinked, hypertext documents that runs over the Internet. We assume the whole World Wide Web as a set of available content on computer systems with a unique IP address at a fixed time. We call our set the Web. The first drawback in every research on this set is that it does change significantly over time and is not predictable. We want to access as much as possible of this set at different times, because it is not possible to dump the Web at a particular point in time. Therefore we crawl content and extract as much links as possible from this content and try to follow these as far as useful. Nevertheless, even such an approach is unlikely to find all malicious content on the Web. Another drawback of our approach is our slow content analysis: compared to the crawling part, our content analysis takes far more time. Signature-based scanning is fast, but it has the limitation of not being able to detect new threats in a timely manner. On the other hand, the behavior-based analysis with CWSandbox takes a longer amount of time. In the future, we need to find ways to analyze the collected content more quickly. The main conclusion for the future work is to use behavioural tactics to find suspicious candidates without a full analysis of their behaviour and to perform a deeper analysis of them afterwards.

4 Results

We have evaluated our system over a period of three months from February until April 2007. In the beginning of February, we have done an initial test crawl with an adult based topic search. After the test crawl, we performed several topic and blacklists-based crawls in March and April 2007.

We have used a commercial off-the-shelf machine utilizing an Intel Pentium 4 CPU with 3.00 GHz and 2048 KB L2-cache. It has a hyperthreading enabled standard processor equipped with 2 GB RAM. The system has three hard disks: one with 80 GB and two with 750 GB capacity. The 80 GB disk is used for the operating system. For the huge storage needs of the crawls and analyses, the two 750 GB disks were grouped together as a logical volume with 1.5 TB. The operating system is a standard Linux distribution.

We chose ClamAV as a malware detector for our evaluation system, since this was the only module with full functionality at the time of our experiments. In contrast to other projects like the UW Spycrawler [MBGL06], we have scanned every crawled content, regardless of its type. This procedure assures us not to miss any malicious content. The drawback of this approach is a performance trade-off for scanning unnecessarily content. In contrast to other anti-virus solutions, the ClamAV anti-virus scanner has additional detection capabilities for phishing and Trojan downloading websites. Additionally, we have observed that some portion of executables, possibly malicious ones, are neither marked as executables in their mimetype specifications nor have a corresponding file extension, e.g. "`.exe`" or "`.msi`". Depending on these advantages, we decided to agree to the slower performance of the system and scan all crawled content.

4.1 Crawls

A First Test Crawl. We have done a first test crawl based on adult search topics in February 2007 with a starting seed of 2205 URLs. We performed this crawl to revise our settings and optimize the crawler performance. During our crawl, we observed some performance bottlenecks depending on the configuration of the crawls. The count of maximum available *Toe-Threads* and the total amount of maximum usable memory had to be adjusted. Toe-Threads are the active crawl threads and responsible for checking, crawling, and analyzing a URL. The initial setting for this option was 250. This value is sufficient with about 250 MB of usable main memory occupied from the Java Virtual Machine. We adjusted the possible memory load for the virtual machine to 1,500 MB and found 600 `Toe-Threads` as a good trade-off for our configuration. We achieved an average download rate of about 1 MB/sec and crawled on average 26 documents/sec.

The Extractor modules which are responsible for extracting URLs from different sources within a web page work relatively efficient. Table 1 shows the extraction efficiency for the initial crawl. We see that Heritrix is able to extract links from a wide range of content types and especially HTML pages contain many links that we can use for our crawl. We need to consider that only discovered links which do pass the requirements are queued for

crawling. These requirements do contain, among others, whether a URL is already queued or downloaded, if a URL is still unique after normalization, and if a URL is in the scope of the crawl.

Module	handled objects	extracted links	links / object
ExtractorHTTP	6,487,350	735,538	0.11
ExtractorHTML	4,132,168	345,231,030	83.55
ExtractorCSS	19,881	88,307	4.44
ExtractorJS	21,658	162,291	7.49
ExtractorSWF	17,921	11,117	0.62
ExtractorXML	35,165	1,638,260	46.59
ExtractorURL	6,506,489	6,776,544	1.04
ExtractorUniversal	1,901,858	1,205,772	0.63

Table 1: Efficiency of different extractor modules for Heritrix

Topic Based Crawls. After the optimizations, we performed five topic-based crawls during March and April 2007. We used the topics ”adult”, ”pirate”, ”celebrity”, ”games”, and ”wallpaper”. To generate starting seeds for these crawls, we used synonymously and commonly used terms regarding the topics utilizing Google Suggest and Google Trends. For example, the topic ”wallpaper” was populated with keywords like ”free wallpapers”, ”wallpapers” etc.

Blacklists Crawl. We aggregated blacklists files from six major blacklist providers in one seed file, which had a total of 8,713 URLs in April 2007.

4.2 Performance

Table 2 shows the seed count per topic, the discovered links, the queued links, and the actual downloaded links. We have observed on average a total rate of 6.48 discovered and queued links per crawled content.

topic	adult	pirate	celebrity	games	wallpaper	blacklist
initial seeds	1,867	2,324	1,405	2,367	2,781	8,713
links found	10,566,820	21,592,129	20,738,435	33,600,204	23,943,186	19,159,246
links queued	8,931,291	18,875,457	17,542,649	27,893,560	18,860,725	15,725,376
downloaded	1,537,145	2,547,330	2,855,300	5,215,394	4,646,027	3,204,560

Table 2: Seed count per topic

Our crawl achieved an average analysis ratio of 0.048 seconds per downloaded content and

an average analysis ratio of 2.35 seconds per downloaded and compressed MB. Table 3 shows the runtime of the malware analysis regarding the runtime of the analysis program.

topic	secs	hours	content count	MB	content	MB/sec
pirate	116,416	32.34	2,547,330	54,182	0.0457	2.15
celebrity	138,2463	38.4	2,855,300	41,877	0.0484	3.3
game	279,741	77.71	5,215,394	168,713	0.0536	1.66
wallpaper	224,220	62.28	4,646,027	150,598	0.0483	0.92
blacklist	138,727	38.54	3,204,560	60,423	0.0433	3.71
total	897,352	249.26	18,468,611	475,795	0.0479	2.35

Table 3: Malware analysis performance per crawl

Figure 2 provides an overview of the mimetype of the downloaded content. More than half of the files are HTML files, followed by images. Another interesting information is the third rank of the `no-type` type depicting a lack of mimetype information in the response. This is an additional endorsement for our approach of malware scanning of all downloaded files rather than only considering executable files, e.g., `application/octet-stream`.

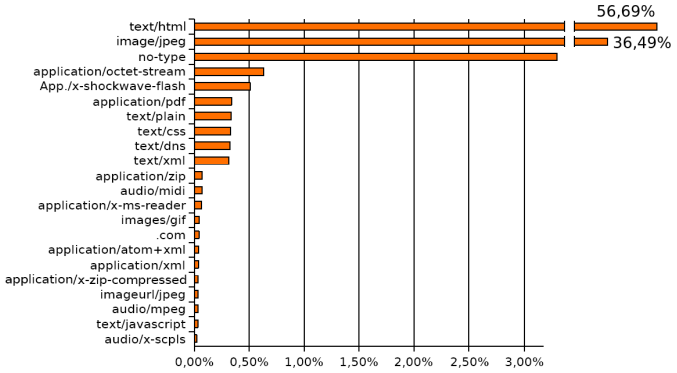

Figure 2: Filetype distribution

4.3 Malicious Websites Results

We know only little about the distribution of malicious websites on the World Wide Web. But our results show an interesting topic-specific maliciousness probability regarding binary files. Table 4 shows the number of malware binaries found per crawling category. Between 0.1 and 2.6 of all found binaries were classified by ClamAV as being malicious and on average 1 percent of all downloaded binaries were malware of some form.

This result can be compared to the result of the SiteAdvisor malware analysis report from March 2007 [DN07]. They identified 4.1 percent of all websites to be malicious. The

topic	maliciousness in %
pirate	2.6
wallpaper	2.5
games	0.3
celebrity	0.3
adult	0.1
blacklist	1.7
total	1.0

Table 4: Topic specific binary file maliciousness probability

gap between their result and our result can mainly be explained with the following two differences: First, SiteAdvisor has additional maliciousness indicators, e.g., drive-by-downloads, and the spam mail analysis. This approach leads to a higher maliciousness rate than ours. Second, users of the SiteAdvisor system can report additional pages to be analyzed. Therefore it is likely that the SiteAdvisor system will discover malicious websites which are not linked on any other site and thus can not be found with our approach.

We have discovered a total of 93 different malware types on more than 50 unique domains during our crawl. Figure 3 provides an overview of the top ten malware samples and malicious domains found during our crawls. Trojan Horses and Adware dominate the collected malware samples and sites offering free downloads and games host this content.

Malware Name	Count
HTML.MediaTickets.A	487
Trojan.Aavirus-1	92
Trojan.JS.RJump	91
Adware.Casino-3	22
Adware.Trymedia-2	12
Adware.Casino	10
Worm.Mytob.FN	9
Dialer-715	8
Adware.Casino-5	7
Trojan.Hotkey	6

(a) Top 10 malware types

Domain Name	Count
desktopwallpaperfree.com	487
waterfallscenes.com	92
pro.webmaster.free.fr	91
astalavista.com	15
bunnezone.com	14
oss.sgi.com	12
ppd-files.download.com	12
888casino.com	11
888.com	11
bigbenbingo.com	10

(b) Top 10 malicious domain distribution

Figure 3: Overview of malicious content found during crawls

Manual URL analysis revealed some false positives and other suspicious domains. False positives can be generated when a benign site hosts examples of attacks, e.g., `oss.sgi.com` contains some binaries that ClamAV identified as malware. Suspicious domains show some unusual activity, e.g., `pro.webmaster.free.fr` seems to have a kind of a so called *spider trap*. This is a mechanism to trap Web spiders / Web crawlers. Such sites automatically generate new URLs on the fly to bother a Web crawler with a needless task.

5 Conclusion and Future Work

In this paper, we introduced the Monkey-Spider project to search for malicious content on the World Wide Web. Using a web-crawler and automated content analysis, we can search for malicious sites within the WWW and examine several properties of this class of attacks against end-users.

Compared to the UW Spycrawler [MBGL06] system, which crawls a fixed number of URLs on specific hosts and crawls the whole content of a host, our analysis is broader: we performed broadscope crawls and ended them knowing that our system could neither store nor process as much data. With this approach, we have seen that the crawler gets mainly HTML sites and focuses on the extraction, but not on the download of the whole content. This leads to only few executable files and thus to a few malicious files. Nevertheless, our experiments show that such an approach is viable and can help to find malicious websites. Moreover, the crawling approch is much faster than high-interaction honeyclients.

Misspelled domain names of popular domains, so called *typos* (typographical errors), are an important source for maliciousness on the Internet. Normally, such domains do not have a meaningful content, except for a coincidental match with other benign sites. The practice of using typos to misuse such shade traffic is commonly known as *typosquatting*. Malware writers want to spread their code as fast and as wide as possible, thus they use general typos of real existing sites and use this to infect innocent users [Web07, FS05]. In the future, we plan to add support for typosquatting domains in Monkey-Spider.

A problem regarding duplicate crawling is the problem of recrawling top ranked websites on every crawl. When we perform broad crawls, top ranked websites like Wikipedia, Amazon, and YouTube, are most likely to be linked frequently on common websites. Thus these sites will most likely be recrawled. These sites do falsify our topics based research results to a certain extent, in giving more ”clean” websites than there should be. Writing a new processor module for Heritrix which can exclude given top ranked sites is our solution to this problem. In this approach, we propose to generate manually or automatically a top rank site list for example out of the Alexa Top Sites [ale] list, and use this list as an exclude rule with a blacklist-like processor. This blacklist processor should avoid crawling contents from any link hosted on the blacklisted top ranked servers. Furthermore, we could additionally crawl only the top ranked site list to gather information about their maliciousness and examine whether they host malware at all.

We conclude that after many years of server security research we need to concentrate more on client security. With the rise and abuse of botnets [FHW05, RZMT06, CJM05], the danger has become greater than ever to be abused by an attacker. The continuance of the Web as we are used to, is now at stake if the power of decomission of network nodes is in the hands of a bunch of cyber criminals controlling botnets and maintaining malicious websites. An easy way of avoiding the threat of malicious websites has to be the automatic avoidance of these dangerous parts of the Web in an easy to use and error-free manner. SiteAdvisor is an example for an approach utilizing the power of the Internet community but is not free and sufficient for the needs. We hope that Monkey-Spider can also be used to collect information that can in turn be used to protect the Internet community.

References

[ale] Alexa Top 500 rank. http://www.alexa.com/site/ds/top_sites?ts_mode=global&lang=none.

[ARC] The ARC file-format. http://www.archive.org/web/researcher/ArcFileFormat.php.

[CJM05] Evan Cooke, Farnam Jahanian, and Danny McPherson. The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets. In *Proceedings of SRUTI'05*, pages 39–44, 2005.

[DN07] Shane Keats Dan Nunes. Mapping the Mal Web, 2007. http://www.siteadvisor.com/studies/map_malweb_mar2007.html.

[FHW05] Felix Freiling, Thorsten Holz, and Georg Wicherski. Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks. In *Proceedings of 10th European Symposium On Research In Computer Security (ESORICS05)*. Springer, July 2005.

[Fra04] Franklin. Follow the Money; or, why does my computer keep getting infested with spyware?, 2004. http://tacit.livejournal.com/125748.html.

[FS05] F-Secure. Googkle.com installed malware by exploiting browser vulnerabilities, 2005. http://www.f-secure.com/v-descs/googkle.shtml.

[her] Heritrix, the Internet Archive's open-source, extensible, web-scale, archival-quality web crawler project. http://crawler.archive.org/.

[IA] Internet Archive. http://www.archive.org.

[Iki] Ali Ikinci. The Monkey-Spider project. http://monkeyspider.sourceforge.net.

[MBGL06] Alexander Moshchuk, Tanya Bragin, Steven D. Gribble, and Henry M. Levy. A Crawler-based Study of Spyware on the Web. In *Proceedings of the 13th Annual Network and Distributed Systems Security Symposium (NDSS 2006)*, San Diego, CA, February 2006.

[McA] McAfee. SiteAdvisor. http://www.siteadvisor.com/.

[Mik] Mike's Ad Blocking Hosts file. http://everythingisnt.com/hosts.html.

[PMM+07] Niels Provos, Dean McNamee, Panayiotis Mavrommatis, Ke Wang, and Nagendra Modadugu. The Ghost in the Browser: Analysis of Web-based Malware. In *Proceedings of HotBots 2007*, 2007.

[Pro04] The Honeynet Project. *Know Your Enemy, Second Edition: Learning about Security Threats (2nd Edition)*. Pearson Education, 2004.

[RZMT06] Moheeb Abu Rajab, Jay Zarfoss, Fabian Monrose, and Andreas Terzis. A multifaceted approach to understanding the botnet phenomenon. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, pages 41–52, New York, NY, USA, 2006. ACM Press.

[Spi02] Lance Spitzner. *Honeypots: Tracking Hackers*. Addison-Wesley Professional, September 2002.

[Ver03] Dan Verton. Internet fraud expanding, security experts warn, 2003. `http://www.computerworld.com/securitytopics/security/cybercrime/story/0,10801,78551,00.html?SKC=cybercrime-78551`.

[WBJ+06] Yi-Min Wang, Doug Beck, Xuxian Jiang, Roussi Roussev, Chad Verbowski, Shuo Chen, and Samuel T. King. Automated Web Patrol with Strider HoneyMonkeys: Finding Web Sites That Exploit Browser Vulnerabilities. In *NDSS*, 2006.

[Web07] Aaron Weber. Amusement Park Typosquatters Install Toolbar via Exploit, 2007. `http://blog.siteadvisor.com/2007/04/amusement_park_typosquatters_i.shtml`.

[WHF07] Carsten Willems, Thorsten Holz, and Felix Freiling. CWSandbox: Towards Automated Dynamic Binary Analysis. *IEEE Security and Privacy*, 5(2), 2007.

	Honey-Monkey	UW Spycrawler	SiteAdvisor	MITRE Honeyclient	HoneyC	Sheila	Capture - HPC	Pezzo-navante	Monkey-Spider
classifi-cation	high-interaction	high-interaction	high-interaction	high-interaction	low-interaction	high-interaction	high-interaction	high-interaction	low-interaction
toolkit availability	proprietary/ not available	proprietary/ not available	proprietary/ not available	Free Software	Free Software	n/a	Free Software	proprietary/ not available	Free Software
infection vector	Web client (IE)	Web client (IE/Mozilla)	Web client (any)	Web client (IE)	Web client (crawler)	email	Web client (various)	Web client (IE/Mozilla)	Web client (crawler), email
malware analysis	Strider Tools FDR, GB, GK	Lavasoft AdAware	n/a	self	Snort	self	state based	many free and proprietary tools	ClamAV
starting year	2005	2005	2005	2004	2006	2006	2006	2005	2006
developer/s	Microsoft Research	University of Washington	McAfee	MITRE	University of Wellington	Vrije Universiteit Amsterdam	University of Wellington	Danford/ Still Secure	University of Mannheim
development status	production/ stable	production/ stable	production/ stable	development/ beta	development/ beta	development/ beta	development/ beta	n/a	development/ beta
analyzed threats	drive-by-downloads, zero-day exploits, malware	spyware	drive-by-downloads, malware, spam	drive-by-downloads, zero-day exploits	malware	malware, zero-day exploits	malware, zero-day exploits, drive-by-downloads	malware, zero-day exploits, drive-by-downloads	malware, spam, phishing
analysis spectrum	Windows OS	Windows OS	Windows OS	Windows OS	multi OS	multi OS	multi OS	Windows OS	multi OS

Table 5: Comparison of different honeyclients