Chappie Swarm: Persona-Driven Web Corpus Generation

Nicholas Kaufman
Noblis

Michael Collins
Redjack

Kristof Ladny
Booz Allen Hamilton

Jeffrey Wiley
Noblis

Adam Plattner
Noblis NSP

Mark Sanders
Noblis

Evan Thaler
Noblis

Patrick Ball
Booz Allen Hamilton

Abstract

A common issue amongst security researchers is the lack of publicly available network traffic traces. In this paper we present Chappie Swarm, which seeks to emulate human behavior in regard to internet browsing. The experimenter can unleash a number of automated chappies which will assume pre-defined personas, and then actively go out and query websites while simultaneously recording their browsing behavior, and saving the network trace as a packet capture file. Unlike other traffic generators, Chappie Swarm distinguishes itself fundamentally by utilizing this "persona" approach, while also not needing to be "seeded" by a previously recorded traffic capture.

1 Introduction

Chappie Swarm is a tool for creating accurate and current websurfing corpora in order to support security experiments. Cyber security experimentation requires access to quality data sets which reflect both attacks *and* normal activity. To this end, security experimentation relies heavily on corpora such as the 1999 Lincoln Labs data set [19], and the KDD Cup dataset [1]. While these sets have many recognized weaknesses [22, 21, 5], they are extensively used in the absence of alternatives.

Chappie Swarm is designed around the principle that corpora must be created and disseminated *quickly*. We contend that any web corpus is affected by problems of external validity (ecological and population validity) [7], producing results that rapidly lose generality.

Researchers have repeatedly noted that network traffic suffers from a significant "moving target" problem [11, 12]. However, web technology arguably changes *faster*, with support technologies such as OAuth(2006), Mobile browsers (2008), AJAX (2005) and HTML5 (2014), resulting in radically different sequences of HTTP requests. This problem also affects aggregate queries; a single multimedia webpage (such as the homepage of CNN) is created through dozens of queries to advertising sites, user tracking, authentication servers, and multimedia caches. This rapid pace of change means that the ecological validity of a corpus may be seriously challenged by the time the corresponding paper is published.

Challenges to population validity come because of the changes between client/server relationships. Modern web traffic heavily relies on the client's identity to provide tailored responses. One example of this tailoring includes geolocation dependent services, notably content delivery networks [15] and HTML5. The increased use of embedded browsing in conjunction with the use of HTTP as a "swiss army knife" protocol has also led to increasingly diverse web clients, including mobile devices, email clients and games. This diversity requires that a corpus be collected from multiple global locations and reflect a diversity of clients.

Our solution is to make the process of corpus generation systematic and reproducible, enabling researchers to quickly create and distribute web corpora reflecting modern Internet architectures. To do so, Chappie Swarm executes surfing based around *personas*. A persona is a Markov process representing common surfing behaviors with preferences for particular websites (*e.g.*, a persona might be interested in news or auctions, while also occasionally checking a sports website). These simulated users then go out and browse the internet in real time. An automated traffic capture is setup, so that a network trace file is automatically generated at the end of each browsing run.

Chappie Swarm enables researchers to easily generate web traffic for any number of users, for any length of time, ensuring that the traffic captured is current. This data may then be published as a corpus, shared with anyone, mitigating the risk of leaking personal information. By scaling up or down the number of agents used, researchers can build corpora addressing different validity challenges.

The remainder of this paper is structured as follows. §2 discusses the architecture of the system. §3 discusses our evaluation approach, and in particular how our system attempts to emulate normal users. §4 discusses previous work, while §5 concludes the work.

2 Chappie Swarm Architecture

In this section, we discuss the architecture and design decisions in building Chappie Swarm. This section is divided into two subsections. §2.1 discusses the core architecture for the system, including the components of the architecture and how they are connected. §2.2 discusses the surfing logic used to emulate humans browsing the web.

2.1 Core Chappie Architecture

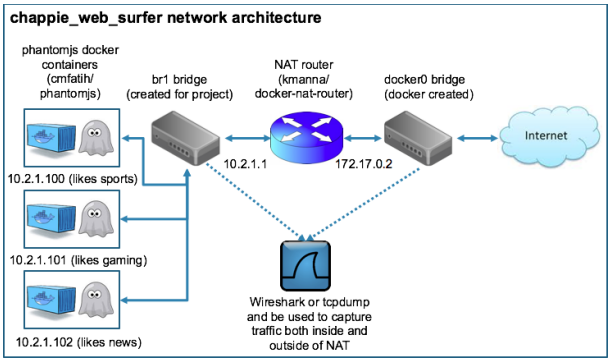

Figure 1: Chappie Swarm Architecture.

Figure 1 shows the Chappie Swarm architecture. As this figure shows, the system consists of 3 major components:

- *chappies*. A chappie is a containerized application containing a virtual web browser and software for executing a *persona*.
- *bridges*. Internal bridges within the Chappie Swarm virtual network that direct traffic.
- *data capture*. A `tcpdump` application collecting traffic at multiple points.

Each chappie is a containerized application launched via Docker. When instantiated, each chappie connects to a virtual network using a common virtual router. Using containers, experimenters can widely distribute chappies across the Internet.

The original experiments motivating Chappie Swarm's development were focused on the impact of NATting on on-the-wire traffic analysis. On most networks any individual host is behind at least one NAT [2]; if network data is collected near a border, it is completely possible that the traffic is hidden in multiple layers of NATting. In §3 we will discuss the impact of NATting in more depth, in this section we focus on how NATting is implemented in the Chappie Swarm architecture.

The virtual network consists of two bridges that grant the experimenter dual vantage on the captured traffic. One bridge captures inside the router, where the experimenter has access to the private information of the chappies, such as IP addresses, port numbers, etc. The second bridge captures from outside the router, enabling the experimenter to simulate NATs. From this external vantage, IP addresses and ports are manipulated by the NAT. During configuration, the experimenter has the option to implement different types of NATs, currently from four variants as outlined in [13] - Symmetric, Full Cone, Restricted Cone, and Port Restricted Cone. In it's current implementation, the purpose of the bridges is to serve as a vantage for traffic capture. Envisioned future usage is to serve as a generic "middlebox." This way, the experimenter will have the flexibility to study the influence network devices have on traffic flow, capture and content.

When the application is started, tcpdump is utilized to capture packets from both the internal perspective as well as the external perspective. After the chappie surfing has concluded, the network traces are automatically shunted to a specified directory for later processing. As of now, the labelling of the generated datasets is fairly minimal. From the "pre NAT" capture, categorical attributes are added based on the uniquely reported IP addresses, to distinguish the traffic on an individualistic basis. Nothing is added to the "post NAT" capture, as that should remain completely anonymized. In order for the chappies to interface with the web, a headless browser is utilized; in the current implementation, this browser is PhantomJS[1].

2.2 Chappie Surfing Logic

In this section, we discuss the chappie *surfing logic*, the process by which individual chappies choose webpages. Our surfing logic is based around two core design principles: the first is, wherever possible, simulating user behavior rather than the internet, the second is the use of personas to guide web choice.

[1] `http://www.phantomjs.org`

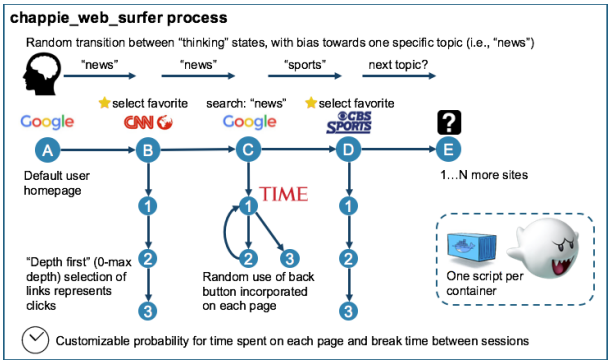

Figure 2: Graphic illustrating the "thought process" of the chappies.

Chappies are designed to use web browsers rather than directly execute requests. The current Chappie Swarm implementation uses PhantomJS, a common headless browser used for web scraping and website testing. PhantomJS, among other features, enables us to automatically execute scripts with multiple `user-Agent` strings. Using PhantomJS, we can simulate actual browsing by feeding it a sequence of websites.

The sequence of websites fed to the browser is determined via the use of a *persona*. Personas are a Markov model based approximation of a user's decision process; in comparison to the random surfing model [8], personas are used to enable chappies to demonstrate differentiable personalities, more accurately imitating human surfing. This emulation is done through the creation of keyworded topic lists about various potential interests.

Table 1 shows the fields that comprise a persona. As this table shows, personas have multiple attributes governing surfing behavior such as the probability of utilization of the browser's back button, how many hyperlinks a chappie should follow from a particular web page, length of time to take a hiatus from browsing, and others. Also in this table are the persona's major *interests*. Each persona is seeded with a base interest, represented by topics such as "video games", "news", "work", etc. Each topic is associated with a corpus of keywords consisting of popular terms scraped from websites, as well as manual additions and synonyms. Within each of these topics, a favorites list is built which the chappies use while navigating the web.

Figure 2 shows how a chappie surfs through a site. As this figure shows, a chappie begins web browsing from a designated homepage. After visiting each page, the chappie will decide whether or not to change their topic of interest, with a probability depending on their current state. Figure 3 gives an overview of this process. If a chappie decides to change their topic of interest, the chappie will utilize their default search engine (another configurable element) to search for a term chosen from the associated corpus for this new topic. Otherwise, the chappie will elect to either click a link from the current web page, navigate directly to another page - simulating the use of the address bar - or to search for a term found in the associated corpus for their current topic.

Figure 3 shows how chappies change topics while surfing. As this figure shows, when a chappie decides to navigate away from its current page, if it elects to click a link found on the current page, a simple text analysis of each link is performed. Weights are then given to each link that is available to the chappie, and the chappie chooses a particular link according to some previously established probability distribution, modified to reflect these weighting changes. The chappie can shift between major and minor topics, or opt to "take a break" from surfing.

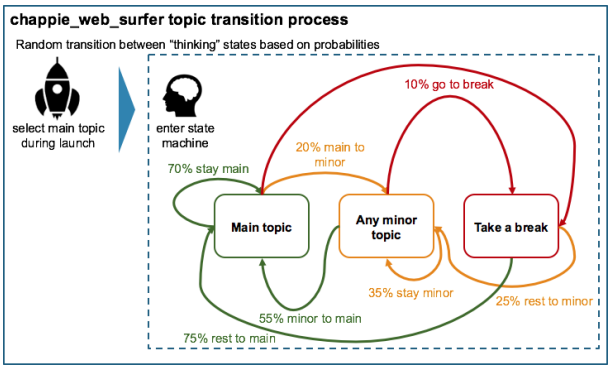

Figure 3: Graphic illustrating the transition diagram between topic states. Percentages listed may be interpreted as example probabilities for transitioning between any two given states.

The personas of each chappie, as well as the characteristics governing browsing behavior (mentioned above) are selected from a config file. In this way, the researcher may easily alter parameters, probability distributions, and add their own custom features. A typical setup is to specify a particular probability array for a given parameter. For instance, one could launch the Chappie Swarm application and specify that the experiment should consist of three chappies, and that the persona for each chappie should be sampled from the pre-configured persona list under a uniform probability distribution, or any desired custom probability distribution.

Chappie Persona Details	
"Major" Topic	{"Work", "Video Games", "News", "Celebrities", "Sports", "Cyber Security"}
"Minor" Topic	{"Major" Topic} \ {Chosen Major Topic}
Search Pages	{google.com, yahoo.com, bing.com, duckduckgo.com}
"Back Button"	3-8% of navigation (default)
Click Emulation	35% of navigation (default)
Search	35% of navigation (default)
Select Favorite	22-27% of navigation (default)

Table 1: Table showing default customization of chappies. Omitted topics include keyword corpora and homepages. For the listed sets, the default behavior is to choose randomly utilizing a uniform distribution. Percentages listed are sample ranges which can be specified during configuration.

3 Chappie Swarm Evaluation

In this section, we describe quantitative measures by which we gauge the realism of the HTTP traffic generated by the Chappie Swarm. This section has three subsections. In §3.1 we discuss our work on trying to emulate accurate user behavior. In §3.2 we compare virtual traffic with real traffic. Finally, in §3.3, we discuss use cases for Chappie Swarm.

3.1 Emulating Human Behavior

Among other considerations, we concern ourselves with a few *key* features.

1. Time between successive page calls
2. Reported User Agent string
3. Types of web pages being visited
4. Http status codes of visited pages

A major concern in evaluating how the chappies behave in terms of realistic web browsing is the time a chappie spends on a particular web page. A common behavior of headless browsers is to visit a large number of pages in a short time, due to the way in which they access DOM elements, and noting that their functionality is often geared towards tasks such as web crawling. Therefore, we implemented a simple throttling mechanism, so that a chappie will spend a (configurable) minimum amount of time on any given web page. This ensures that we can emulate a human reading an article, interacting with a website, etc. Further, the amount of time spent on a particular page is not a set value, but randomly sampled according to a (configurable) probability distribution.

Another aspect of traffic generation that was of concern to the authors was the reported user agent string. This is easily configured, and the authors currently assign a user-agent string to a chappie using a uniform probability distribution from a list of the ten most popular user-agent strings according to `useragentstring.com`, a repository of user agent strings for all web browsing platforms.

One of the larger concerns of realism is the types of web pages being selected by the chappies. For instance, after performing a Google search for a topic, often the top several results are advertisements. It's a well studied problem that the click rate of advertisements is exceedingly low [10] - in fact, bot-nets are sometimes built with the primary functionality of boosting revenue through ad clicks - and thus it does little to address the problem of realism to incorporate a functionality such as searches if the default behavior is to click links that most humans would intentionally avoid. To this end, the chappies have been designed to avoid all ad-links. Further, we have implemented a more sophisticated system of deciding which website to visit, given a list of web links extracted from a specific page. Each "topic of interest" that a chappie might adopt has associated with it a corpus of "key words." The links on a given web page are parsed, and weights for each of these links are adjusted based on the degree of overlap with the words in the URL and the words in the "key word" corpus.

3.2 Comparing Virtual Traffic with Actual Traffic

In an effort to validate that the Chappie Swarm produces realistic looking traffic, we compared the statistical footprint of the chappies with that of actual humans. To this end, we employed two people to record their own web traffic for fixed time intervals. All of these packets were run through a protocol de-multiplexor, Bro, and compared. With the caveat that human behavior is highly variable and dynamic, we note that timing between web pages loaded, status codes returned, and number of unique destination IP addresses visited are comparable. Figures 4 and 5 show comparisons of the two former items from a particular trial.

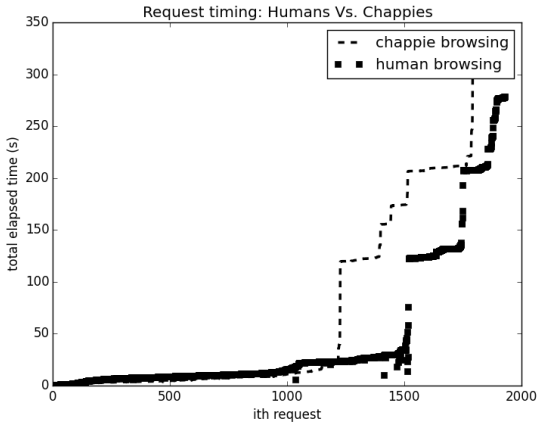

Figure 4: Plot shows the difference in timing between a human and a chappie when making browser requests. In this instance, a human was given a "script" to follow, based on the chappie's pre-recorded browsing. Timing comparison shows the ability of the human to nearly recreate the "script", and their attempts at recreation without knowing the full details of the "script."

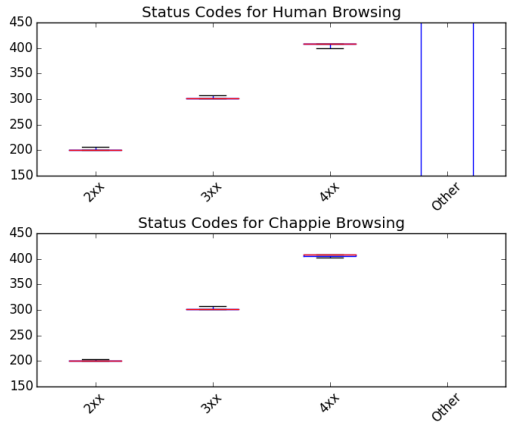

Figure 5: Plot shows the difference in status codes between a human and a chappie when making browser requests. This is taken from the same "script" as Figure 4.

3.3 Specific Use Cases

Here we discuss specific use cases for Chappie Swarm. The first case involves identifying distinct web sessions, the original motivation. The second case shows how chappies can be used to address problems in population validity due to geolocation.

3.3.1 Reconstructing Webpages Across a NAT

The original motivation for developing Chappie Swarm was in order to study the impact that common traffic management tools, particularly NATting, have on a network monitor's ability to reconstruct web sessions. This work sought to extend [2]. Webpages often consist of dozens or hundreds of constituent requests, making it difficult to rely simply on timing or addresses to determine where one web page ends and the next begins. NATting further complicates the problem by merging multiple source addresses into a single address.

In order to test this, the Chappie Swarm application was run with three chappies under each of the four NAT protocols as detailed previously. Network traces were captured both internally and externally, so that one could have a notion of "ground truth" labelling, as well as a NATted data set. This data set enables us to study multiple techniques for session reconstruction.

3.3.2 Geographical Profiling

Another potential issue in generating web traffic is the effect of Content Distribution Networks, and the underlying regionality which comes with such utilization. From the standpoint of ecological validity, Chappie Swarm can be utilized to analyze the differences in traffic patterns across websites from different geographic locations. For instance, if a researcher in Northern California were to launch one hundred chappies to browse the web, and a researcher in Florida were to launch one hundred chappies, the resources loaded by websites supported by CDNs will appear to come from very different locations. Utilization of Chappie Swarm in conjunction with a service such as Amazon Web Services (AWS) would allow a researcher to have a better understanding of the variance associated with a particular website.

4 Previous Work

Previous work in security experimentation has depended heavily on corpus data generated by testbeds. The fundamental example of this work, is the original Lincoln Labs LARIAT dataset [19] and its descendant the 1999 KDD cup [1]. As discussed in §1, these data sets have well-known weaknesses. Other experimenters have used other web corpora, such as traces of user clicks [17]. Our primary concern in developing Chappie Swarm was addressing the problem of currency – ensuring that we had access to web traffic representative of the Internet as needed. To that end, we envisioned and designed Chappie Swarm as a *process* – something that we could rerun as often as necessary.

Web security researchers have created a number of crawling approaches for creating web corpora. Lee *et al.*'s malicious web crawling research [18], and Coull *et al.*'s work on web browsing [9] create corpora by automatically crawling the web. These approaches differ from Chappie Swarm's by focusing on capturing the websites as corpora, while the focus of Chappie Swarm is on capturing the web *traffic*. Other capture techniques involve crowdsourcing web crawling activity, such as the browser plugin created for Lu *et al.*'s SURF system [20], or the Mechanical Turk-based approach used by Bursztein *et al.* [6]. In comparison to these approaches, Chappie Swarm is automated and designed to capture the traffic and the webpages. In this, we have taken cues from Korbar *et al.*'s agent based work [14], as well as Silverman's "realism" approach [24] and Blythe's work in cognitive models [4].

Outside of the security domain, the most directly relevant work is focused on search engine optimization. Standard works on search engine optimization focus on alternative models of surfer behavior, notably the random surfer model [3, 16] and PageRank [23]. These models (particularly PageRank) influenced the chappie design, but these approaches are less concerned with user activity than we are with Chappie Swarm; much of Chappie Swarm's design is focused on emulating user behavior such as delays between clicks and distractions from searching, issues that are not directly relevant to the search engine literature.

5 Conclusions

In this paper, we have described Chappie Swarm, a tool for systematically creating HTTP traffic traces. We have built this system in order to regularly create web traffic traces that reflect the web as it exists currently, as the constant change in web technology directly challenges the validity of any web corpus. In doing so, we have deemphasized the role of simulation in favor of direct interaction with the Internet. We posit that the more technical debt incurred from doing full packet captures is offset by the improvement in external validity.

We highlight several use cases for the collection of HTTP traffic using Chappie Swarm and further show that these traffic traces are statistically similar to those generated by humans under certain conditions. Additionally, we have discussed the large degree of customizability of Chappie Swarm, and show that it can be incorporated in many different network architectures. We believe that the greatest problem Chappie Swarm solves is in the generation of realistic web traffic which can be shared amongst researchers across different organizations and communities, without fear of compromising network security nor personal privacy.

Chappie Swarm is well suited to solving the problem it was originally designed for – testing the validity of web session reconstruction over NATted traffic. Future design work focuses on emulating user behavior more accurately and more precisely. To do so, we will add more advanced modeling to categorize topics outside of major and minor families, improving the mimicry of user behavior, and improving the automation to enable any researcher to reproduce a corpus.

The current persona model is very simple, and chappies run into functionality issues for occasionally visited websites. To ensure a seamless browsing process, we intend to expand the chappie's decision model to include additional categories of sites, including blacklists of sites that a user would *never* visit (*e.g.*, they don't speak the language, radically different opinions). In addition, we intend to extend and automate the topic lists; the current lists are hand-curated. To do this, we intend to use Alexa rankings to classify each website for one or more personas.

Other fidelity issues involve creating a better emulation of user behaviors such as clicking and other website interactions. To do so, we will need to move past the current headless browser design and experiment with other web testing and validation tools such as Selenium[2].

Similarly, we intend to look into improving the quality of web sessions; ensuring that the sequence of pages visited by a chappie "makes sense" in that some semblance of a story can be pieced together by looking at the subsequent links that a chappie visits. Initially, for small data samples, this was done "by sight." As the development of Chappie Swarm proceeded, however, the need for more robust and quantitative measures became obvious. Such work will involve extensive user testing.

Longer term goals include automating the process of surfing and generation so that Chappie Swarm scripts can be shared among research groups. In this way, the best practices for generating a corpus can be shared. Ideally, projects using Chappie Swarm to generate web corpora will include as part of their release the collected data, and the configurations used to generate that data.

References

[1] ARCHIVE, U. K. Kdd cup 1999 data. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[2] BELLOVIN, S. M. A technique for counting natted hosts. *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment* (2002).

[3] BLUM, A., CHAN, T. H., AND RWEBANGIRA, M. R. A random-surfer web-graph model. In *Proceedings of the eigth Workshop on Algorithm Engineering and Experiments and the third Workshop on Analytic Algorithmics and Combinatorics* (2006).

[2] http://www.seleniumhq.org

[4] BLYTHE, J. A dual-process cognitive model for testing resilient control systems. International Symposium on Resilient Control Systems (ISRCS), pp. 8–21.

[5] BRUGGER, T. Kdd cup '99 dataset (network intrusion) considered harmful. http://www.kdnuggets.com/news/2007/n18/4i.html.

[6] BURSZTEIN, E., BETHARD, S., FABRY, C., MITCHELL, J. C., AND JURAFSKY, D. How good are humans at solving captchas? a large scale evaluation. *2010 IEEE Symposium on Security and Privacy* (2010).

[7] CAMPBELL, D., SHADISH, W., AND COOK, T. *Experimental and Quasi-Experimental Designs For Generalized Causal Inference*. Wadsworth Publishing, 2001.

[8] CHEBOLU, P., AND MELSTED, P. Pagerank and the random surfer model. In *Proceedings of the 2008 ACM Symposium on Discrete Algorithms* (2008).

[9] COULL, S. E., COLLINS, M. P., WRIGHT, C. V., MONROSE, F., AND REITER, M. K. On web browsing privacy in anonymized netflows. In *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium* (2007).

[10] FANG, Z., YUE, K., ZHANG, J., ZHANG, D., AND LIU, W. Click-through rate estimation for rare events in online advertising. *Mathematical Problems in Engineering 2014* (2014).

[11] FLOYD, S., AND PAXSON, V. Difficulties in simulating the internet. *IEEE/ACM Transactions on Networking 9* (2001).

[12] GATES, C., AND TAYLOR, C. Challenging the anomalydetection paradigm: A provocative discussion. In *Proceedings of the 2006 NSPW Workshop* (2006).

[13] HUSTON, G. Anatomy: A look inside network address translators. `https://web.archive.org/web/20070220131824/http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_7-3/anatomy.html`.

[14] KORBAR, B., KOPPEL, R., KOTHARD, V., AND SMITH, S. Validating an agent-based model of human password behavior. AAAI Workshops.

[15] KRISHNAMURTHY, B., WILLS, C., AND ZHANG, Y. On the use and performance of content distribution networks. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement* (2001).

[16] KUMAR, R., RAGHAVAN, P., RAJAGOPALAN, S., SIVAKUMAR, D., TOMKINS, A., AND UPFAL, E. Stochastic models for the web graph. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, FOCS '00.

[17] LEE, L., JUAN, Y., LEE, K., TSENG, W., CHEN, H., AND TSENG, Y. Context-aware web security threat prevention. In *Proceedings of the 2012 CCS conference* (2012).

[18] LI, Z., ZHANG, K., XIE, Y., YU, F., AND WANG, X. Knowing your enemy: understanding and detecting malicious web advertising. In *Proceedings of the 2012 CCS conference* (2012).

[19] LIPPMANN, R., HAINES, J. W., FRIED, D. J., KORBA, J., AND DAS, K. The 1999 darpa off-line intrusion detection evaluation. *Computer Networking 34*, 4 (Oct. 2000).

[20] LU, L., PERDISCI, R., AND LEE, W. Surf: detecting and measuring search poisoning. In *Proceedings of the 2011 CCS conference* (2011).

[21] MAHONEY, M. V., AND CHAN, P. K. An analysis of the 1999 darpa/lincoln laboratory evaluation data for network anomaly detection. In *In Proceedings of the Sixth International Symposium on Recent Advances in Intrusion Detection* (2003), Springer-Verlag, pp. 220–237.

[22] MCHUGH, J. Testing intrusion detection systems: A critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM Trans. Inf. Syst. Secur. 3*, 4 (2000).

[23] PAGE, L., BRIN, S., MOTWANI, R., AND WINOGRAD, T. The pagerank citation ranking: Bringing order to the web. Tech. Rep. 1999-66, November.

[24] SILVERMAN, B. G. More realistic human behavior models for agents in virtual worlds: Emotion, stress, and value ontologies. Tech. rep., 2001.