# Flash Crowd Detection Using Decoy Hyperlinks

**Conference Paper** · May 2007

DOI: 10.1109/ICNSC.2007.372823 · Source: IEEE Xplore

Dimitris Gavrilis
**39** PUBLICATIONS **598** CITATIONS
SEE PROFILE

E. Dermatas
University of Patras
**91** PUBLICATIONS **721** CITATIONS
SEE PROFILE

# Flash Crowd Detection Using Decoy Hyperlinks

Dimitris Gavrilis, Ioannis Chatzis and Evangelos Dermatas

*Abstract*— **In this paper a novel method for detecting Denial of Service attacks (DoS) on web services are presented and evaluated by using decoy hyperlinks embedded in web pages. The Decoys are hyperlinks without semantic information or are invisible to the human user, acting like traps for DoS attacks because a human user would never follow them. An attack on a web server is detected when such hyperlink is followed. This approach has significant advantages over other approaches like graphic Turing tests, it is transparent to the user, it can be used on general-purpose web sites and retains the web site's usability. The proposed method has been evaluated using both real and simulated web sites and the results show false positive rates that are less than $10^{-4}$. The aspects of this new method are discussed and some experimental results are presented.**

## I. Introduction

The recent advances of web services and their enrichment with technologies such as Asynchronous JavaScript with XML (AJAX) have led to a turn from classical desktop software to complex browser enabled web applications. As a result, this rapid development of web applications, has led to the creation of new attacks that target such services and applications. Computer attacks targeting web services come in two forms: those that exploit a vulnerability of the web service and those that use legitimate means to exhaust the web server's resources. In the first case, the most popular example is probably SQL injection attacks while in the latter, the DoS attacks that target Web services (Web DoS) is a distinctive example.

The Web DoS attacks are a simple, yet effective form of web attacks that usually targets popular web sites. In this form of attacks, the attacker aims at reducing (or depleting) the effective number of users that can be serviced. To achieve that, the attacker simulates legitimate user navigation with the use of an automated program. Such attacks are not uncommon and sometimes can totally prevent legitimate users to use a web service. The tools to launch such attacks are simple to build and can be very effective. If the attack does not aim at exhausting the server's resources, but instead aims at reducing the server's user capacity, it is extremely difficult to detect it. This is because the server has only limited information for the client (IP address, Timestamp, Web browser, Requested Page, etc.) and there are thousands IP's involved in the attack (when the attack is distributed). Such an attack was made public recently when a Company in Massachusetts paid hackers to launch a Distributed DoS attack against three of its competitors [9]. The attack involved more than 10,000 compromised machines. During the attack, the SYN-flood failed so the attackers used Web based denial of service (Web-DoS or HTTP-flood) by requesting large image files from the victim's servers. The victim's servers remained offline for two weeks. It is worth noting that the attack described here is the simplest form of Web DoS attack that will be presented later in this paper.

Flash crowd events are situations where suddenly the demand for a web site increases rapidly. This can be due to popular news posted at a web site or because of the time (e.g. morning hours at a news portal). A Web DoS attack is very much like a flash crowd event at the server's point of view. Because the server has little information on the clients it is serving, it is difficult to distinguish between a flash crowd event and a Web DoS attack. Furthermore, a false-positive decision (this is when a Flash crowd event is mistakenly treated as a Web DoS attack) could be catastrophic because all the legitimate users would be denied access to the services they requested.

## II. Related Work

Recently, a number of Web-DoS detectors have been presented and can be classified into two categories. The first type of detectors is based on surfer models stored in the Web server log-files. A surfer model is derived from the click streams and is matched against stored models. This approach could be used to classify any Web-activity into two types of user: a human activity and a machine based Web-surfer.

In [2], Mukund and Karypis present three pruning algorithms that are used in reducing the states of $K^{th}$ order Markov models. The authors try to derive user models for web services. They test their method on three datasets and show that the proposed method outperforms both $1^{st}$ and simple $K^{th}$ order Markov models. Ghosh and Acharyya [3] shown that the $1^{st}$ order Markov model is inadequate to describe accurately the user behavior and higher order Markov chains are too complex and computationally intensive. Instead, they propose a method to create and map Web-pages to concept trees which the user traverses. This approach has the advantage that the surfer models are

Dimitris Gavrilis is a PhD candidate at the Electrical & Computer Engineering Department of University of Patras Greece (e-mail: gavrilis@upatras.gr).

Ioannis Chatzis is a PhD candidate at the Electrical & Computer Engineering Department of University of Patras Greece (e-mail: chatzis@upatras.gr).

Evangelos Dermatas is an Assistant Professor at the Electrical & Computer Engineering Department of University of Patras Greece (e-mail: dermatas@george.wcl2.ee.upatras.gr).

derived from the concepts of the pages rather than the pages themselves.

In [4], Ypma and Heskes propose a mixture of Hidden Markov Models for modeling the surfer's behavior. The page topics are extracted from the pages based on the surfer's click streams. Consequently, a surfer type is recognized based on those topics. The authors also incorporate prior knowledge about the user behavior into the transition matrices in order to achieve better results. In [5], an alignment algorithm is employed to the longest repeated sequence to predict the surfer behavior. This approach is suited for web surfer modeling where there are variable length sequences.

In the second category, architectures that are DoS resistant are presented. The authors in [6,7] propose the Secure Overlay Services (SOS), an architecture that hides the server’s true location, providing cover from DoS attacks. Although the proposed architecture does not require modifications on the server or the client, the clients must be aware of SOS. The authors have also used Turing tests to allow only human users to access the SOS services. It is understandable that SOS does not target publicly open services such as commercial Web-sites.

In [8], where the authors present Kill-Bots, a Linux kernel based system designed to detect and block Web-DoS attacks. Kill-Bots combines a two stage authentication mechanism that includes graphical tests and an admission control algorithm that protects the server from overloading.

In this paper a novel detection method of Web DoS attacks are presented and evaluated using both real and simulation experiments. The proposed method uses decoy hyperlinks containing semantic information detectable only by automated programs (programs that launch Web DoS attacks). An example of such information is hyperlinks hidden in an image map that is invisible to a human.

The structure of this paper is as follows: In section III, the types Web-DoS attacks are described in detail. In section IV, the proposed detection method is presented. The experimental evaluation, some simulation results are shown along with some results from a real Web-site and a short discussion is given in sections V and VI.

## III. Web DoS attacks

The Web DoS attacks can be divided into three categories: In Type-I, the same page is requested constantly as shown in Fig. 1. This is the simplest form of Web DoS attack and can be detected using simple thresholds that isolate the IP addresses requesting the same page more than N times per period. The well known tool mod_evasive [9] can combat Type-I attacks.

In Type-II, the attacker first scans the web site and then constantly requests a random web page. This type of attack is shown on Fig. 2 and can be detected by measuring the variance of each click stream and whether sequential page requests are connected. Type-II attacks are more difficult to detect than Type-I and the mod_evasive tool [9] can partially detect and block Type-II attacks.

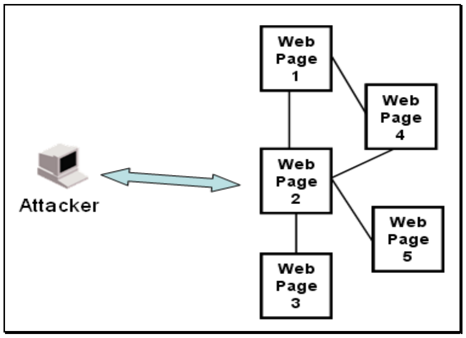

Fig. 1: Type-I Web DoS attack: the attacker constantly requests one of the web site’s pages.

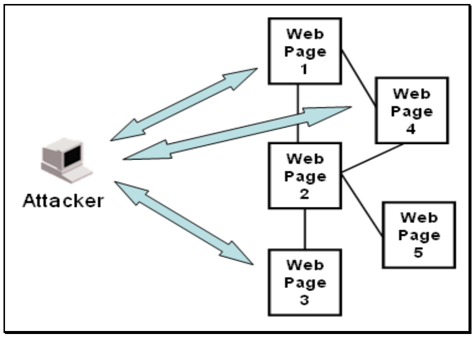

Fig. 2: Type-II Web DoS attack: the attacker constantly requests random pages.

In Type-III, the attacker scans the web site and then randomly creates navigation patterns that resemble real human web surfers. Type-III attacks are the most difficult to detect and a typical example is shown in Fig. 3 where a navigational pattern is displayed with the red arrows. The attacker starting from Page 1, follows a hyperlink from Page 1 to Page 4 and then from Page 4 follows a hyperlink to Page 2.

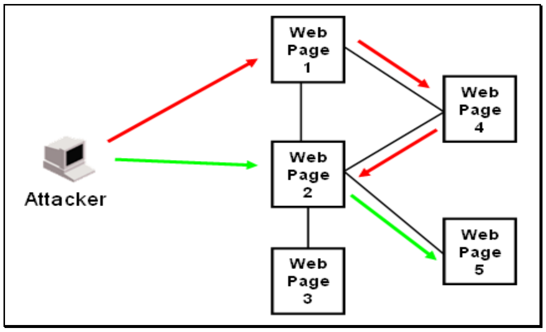

Fig. 3: Type-III Web DoS attacks: the attacker scans the web sites and creates random navigation patterns.

Type-III attacks are extremely difficult to detect especially when they are launched from multiple IP addresses, probably thousands. The server does not have enough information to distinguish a program from a real web surfer. User modeling methods are extremely difficult to implemented in general purpose sites, requiring also adaptive training of user models. Thus, the only effective technique to combat Type-III attacks is the use of Turing tests (Fig. 4) where the user is requested to type a string which has been printed on an image and then has been distorted. Although Turing tests are effective, they have major drawbacks:

- They are difficult to incorporate on general purpose web sites.
- They reduce the web site's usability, especially for general purpose web sites that do not require the user to authenticate.
- Blind users cannot web sites that contain graphic authentication such as Turing tests.

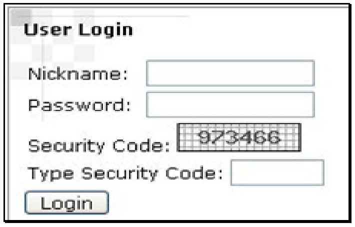

Fig. 4: A graphical authentication (Turing test) where the user is required to type what he/she sees on a distorted image.

The proposed method detects Type-III attacks in a significantly simpler implementation and bears none of the drawbacks that the Turing tests introduce. This method is described in detail in the next section.

## IV. Web DoS Detection

The proposed system for detecting Web DoS attacks consists of 3 modules as shown in Fig. 5. Each module detects one of the three types of Web DoS attacks. An engine decides whether an attack is in progress by scanning the web server's logs and feeds the necessary information to the three distinct classifiers (each one for Type-I, Type-II and Type-III attacks). Based on the results of the three classifiers a simple decision engine decides whether an attack is in progress.

The three classifiers could interact directly with the web server or indirectly through its logging subsystem. Each one of the three modules is described in detail below. Emphasis is given on the Type-III classifier.

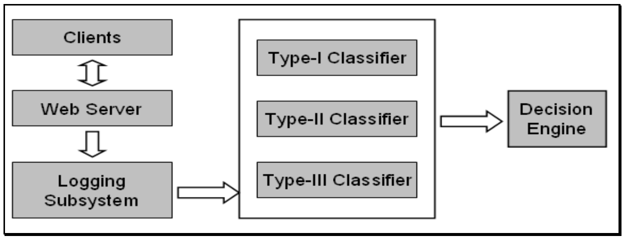

Fig. 5: The proposed system architecture for detecting Web DoS attacks.

### A. Type-I classifier

The Type-I classifier uses finite state machines (FSM) to detect and block attackers that constantly request the same page. An example of such an FSM can be seen below:

```
If (  Count(IP_Addr, URI) > Threshold
&& !Whitelist(IP_Addr)  ) Then Block(IP_Addr);
```

### B. Type-II classifier

The Type-II classifier measures whether sequential pages that are requested in each click stream are interlinked. If the attacker in a Type-II attack requests pages randomly, there is a high probability that sequential pages are not interlinked. The detection of such an event could signal an ongoing Web DoS attack.

### C. Type-III classifier

The Type-III classifier uses decoy hyperlinks embedded in the web pages to distinguish between legitimate users and automated programs. Decoy hyperlinks are especially designed hyperlinks that are invisible to human users. Their invisibility properties are derived either from their semantically indifferent to a human or because the web browser renders the hyperlinks in such way so they are invisible to the human eye. An example for the first case is the image map shown in Fig. 6 that contains two legitimate hyperlinks (the two flags that alter the viewing language) and on decoy that is located in a point on the image map that gives out no information to the human user.

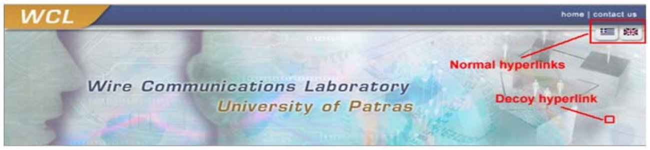

Fig. 6: An image map that contains two normal hyperlinks (the two flags on the top right) and one decoy.

In the second case, a decoy hyperlink could be embedded in a cell of a table where the table itself has been marked as hidden. In this case, the browser would hide the hyperlink, while a simple automated program could not detect the difference. The decoy hyperlink method has three major advantages:

- It is completely transparent to the user.
- It requires minimum modifications on existing web sites. Only few decoys hyperlinks are embedded in certain web pages.
- It does not rely on authentication mechanisms and thus is suitable for general purpose web sites.
- It is an accessible technology in contrast to graphic Turing tests.

A drawback of the detection system that employs decoy hyperlinks is that an attacker could create navigational patterns by him/herself and replicate those patterns at a scale level. In order to overcome this problem, each hyperlink could be accompanied with an extra variable which would act as an access key. This key could be changed periodically (or in cases of high demand) in order cancel already stored navigational patterns. This would force the attacker to use a completely random program.

A white list should also be used in order to allow bots (e.g. googlebot, msnbot etc.) to crawl the web site without raising any alarms.

Taking into account that a typical modern web site consists of a few hundred pages and that the process of inserting decoy hyperlinks in Web-pages is not only cost and time expensive but it also requires extensive testing, two aspects of the proposed system that must be addressed are:

- Construction of decoys in order to minimize false positives.
- Selection of a minimal number of links and pages as decoys in order to maximize the probability a Web-DoS attack will hit them.

### A. *Construction of decoy hyperlinks*

The decoy hyperlinks must be undetectable to human users, while an automated program will not be able to distinguish between them and real hyperlinks. A great number of different types of decoy hyperlinks can be created:

- A few pixels in an image map hidden in some image of the Web-page.
- A hyperlink is invisible to human users if the hypertext has the same color as the page background.
- Hyperlinks without hypertext
- Decoy hyperlinks could also be embedded in hidden tables.

### B. *Selection of decoy Web-pages*

To address this problem, the Web-site is represented by a undirected Graph *G(V,E)*, where the graph's vertices *(V)* represent the site's pages and the graph's edges *(E)* represent the hyper links (the interconnection between the pages). The graph is undirected because the reverse direction of a hyper link can be accomplished using the Back button of any browser. The attacker can be treated as a random walker on the graph. The optimum subset of a concentration of traps $c$ ($c \in [0,1]$) that minimizes the survival probability of the random walker $\phi(n,c)$ is requested. This survival probability depends on the concentration $c$, the number of steps of the random walk $n$ and the trap configuration. The trap configuration is produced by the following trap selection function:

$$V_{trap} = \arg\max_{u}\{d_G(u) \mid u \in V\},$$

where $d_G(u)$ is the degree of the vertex *u* and $V_{trap}$ denotes the vertices that are to be embedded with traps (decoy hyperlinks). This selection function has been experimentally found using simulation experiments [10] and it is a simple but effective means of choosing which pages to embed with decoys. The efficiency of the proposed selection function is compared to the solution is given by a genetic algorithm. The comparison details are discussed in the next section.

## V. Experimental results

The experimental results concerning the proposed method address the two aspects that were described in the previous section.

First, the types of the decoys described in the previous section were embedded in the official web site of the Electrical & Computer Engineering Department of the University of Patras. After a period of one month, 45121 hits were recorded. Only 19 of them were decoy hyperlinks originating from normal users (the rest were various bots). This indicates a probability of less that $10^{-4}$ that a normal user would discover and follow a decoy hyperlink. Furthermore, from the 19 hits, only 3 users clicked on a decoy hyperlink twice in the same session.

Second, to address the selection problem of a minimum subset that maximizes the detection probability, web sites are simulated with a graph and the surfer/attacker was simulated by a random walker. The simulated web sites had the following characteristics:

- Order of 50 pages (vertices).
- Connectivity is 10% between pages (vertices).
- Diameter is 10% of the order |V| of the graph.

The random walker (web surfer or attacker) performs a simple random walk (SRW) starting from a random vertex of the graph and moving with equal probability among the possible routes for N steps: 3, 5, 7, 9. The Web-Dos detection probability is derived from 1000 statistical independent tests for each one of 1000 randomly created web sites (graphs).

As shown in Fig. 7, the detection probability increases as the steps of the SRW raises and most importantly, for the case where the step N of the SRW is set to 9, the detection probability is 0.478 for setting decoys on only 5 vertices (10% of the graph).

The efficiency of the proposed trap selection function is compared to a genetic algorithm whose search space is all the possible random graphs whose decoys cover N percent of the total pages. The genetic algorithm implementation details are:

- Tournament selection
- 0.95 crossover rate
- 0.05 mutation rate
- Population of 100 chromosomes
- Each chromosome is a sequence of binary numbers (0=normal page, 1=decoy).
- Maximum number of generation: 100

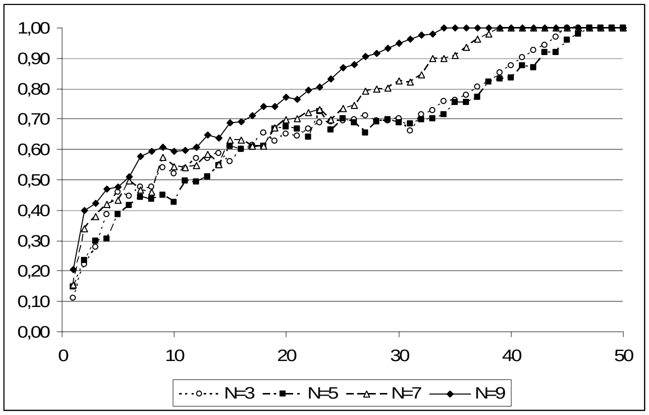

Fig. 7: Detection probability (vertical axis) for a graph of order 50 and decoys varying from 1-50 (horizontal axis).

The Web-Dos detection probability is derived from 1000 tests for each one of 1000 randomly created graphs using the statistical characteristics mentioned above. The experimental results are shown in Fig. 8, where the detection probability (vertical axis) is plotted for different number of vertices containing decoys (horizontal axis) for graphs of order 50. In the worst case, the initial detection difference of 13.9%, measured when only five decoy hyperlinks are embedded in the web site. This difference decreases continuously when the number of decoy hyperlinks increases and converges at 30 vertices with decoys (60% of the graphs total vertices).

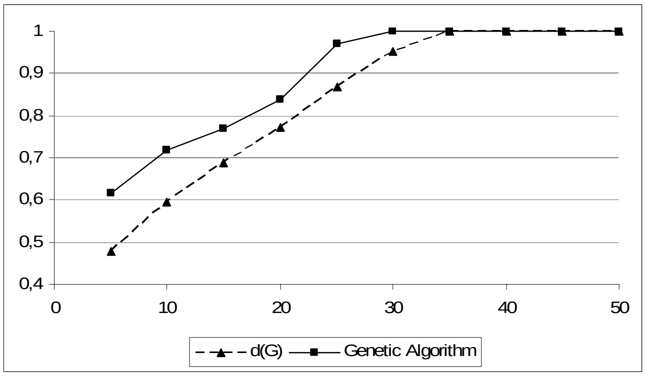

Fig. 8: Comparison of the proposed selection function to a genetic algorithm.

## VI. CONCLUSION & FUTURE WORK

A novel method for detecting flash crowd events has been presented using decoy hyperlinks embedded in web pages to detect Web-DoS attackers. This method has significant advantages over classical Turing tests methods. Those advantages are its simplicity, usability, and its capability of being able to function for general purpose web sites. Experiments on real and simulated web sites have shown that it is possible to create such decoys so that the false positive rate (a human to follow a decoy) is kept near to zero. An algorithm for selecting a relatively small subset of the web sites pages to embed with decoys has also been presented. Experimental results on simulated web sites have shown that a detection rate of 0.577 could be achieved by modifying only 7 web pages (14%) of the web site (for attacks of 9 steps).

Future work includes testing the proposed method on real web sites. Also, the case of creating more sophisticated traps (using JavaScript dynamic hyperlinks) is considered. Finally, another aspect that needs to be examined is the use of dynamic decoys that move from page to page.

## REFERENCES

[1] K. Poulsen, “FBI Busts Alleged DDoS Mafia” , http://www.securityfocus.com/news/9411/

[2] Mukund Deshpande and George Karypis, “Selective Markov for Predicting Web-Page Accesses”, Technical Report #00-056, University of Minessota, 2000.

[3] Acharyya Sreangsu, Ghosh Joydeep, “Context-Sensitive Modeling of Web-Surfing Behaviour using Concept Trees”, in Proceedings of the 5th WEBKDD Workshop, Washington, 2003.

[4] Alexander Ypma and Tom Heskes, “Automatic Categorization of Web Pages and User Clustering with Mixtures of Hidden Markov Models”, in Proccedings in the 4th WEBKDD Workshop, Canada, 2002.

[5] Weinan Wang and Osmar R. Zaiane, “Clustering Web Sessions by Sequence Alignment”, in Proceedings of DEXA Workshops, 2002, pp. 394-398.

[6] William G. Morein, Angelos Stavrou, Debra L Cook, Angelos Keromytis, Vishal Misra, Dan Rubnstein, “Using Graphic Turing Tests To Counter Automated DdoS Attacks Against Web Servers”, Proceedings of the 10th ACM International Conference on Computers & Communications Security, Washington 2003.

[7] D.L. Cook, W.G. Morein, A.D. Keromytis, V. Misra, D. Rubenstein, “WebSOS: protecting web servers from DDoS attacks”, Proceedings of the 11th IEEE International Conference on Networks (ICON), 2003, pp. 455-460.

[8] Srikanth Kandula and Dina Katabi and Matthias Jacob and Arthur W. Berger, “Botz-4-Sale: Surviving Organized DDoS Attacks That Mimic Flash Crowds”, 2nd Symposium on Networked Systems Design and Implementation, Boston, 2005

[9] Jonathan A. Zdziarski, “mod_evasive”, http://www.nuclearelephant.com/ projects/mod_evasive/.

[10] Dimitris Gavrilis, Evangelos Dermatas, “Detection of Web Denial-of-Service Attacks using decoy hyperlinks”, 5th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP), Patras, 2006