(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
4 August 2005 (04.08.2005)

PCT

(10) International Publication Number
WO 2005/069732 A2

(51) International Patent Classification: Not classified

(21) International Application Number:
PCT/IL2005/000081

(22) International Filing Date: 23 January 2005 (23.01.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/539,327 26 January 2004 (26.01.2004) US

(71) Applicant *(for all designated States except US)*: **CISCO TECHNOLOGY INC.** [US/US]; 170 W. Tasmin Drive, San Jose, CA 95134 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **TOUITOU, Dan** [IL/IL]; 21 Golani Street, 52224 Ramat Gan (IL). **ZADIKARIO, Rafi** [IL/IL]; 9 Sokolov Street, 44256 Kfar Saba (IL).

(74) Agents: **SANFORD T. COLB & CO.** et al.; P.O. Box 2273, 76122 Rehovot (IL).

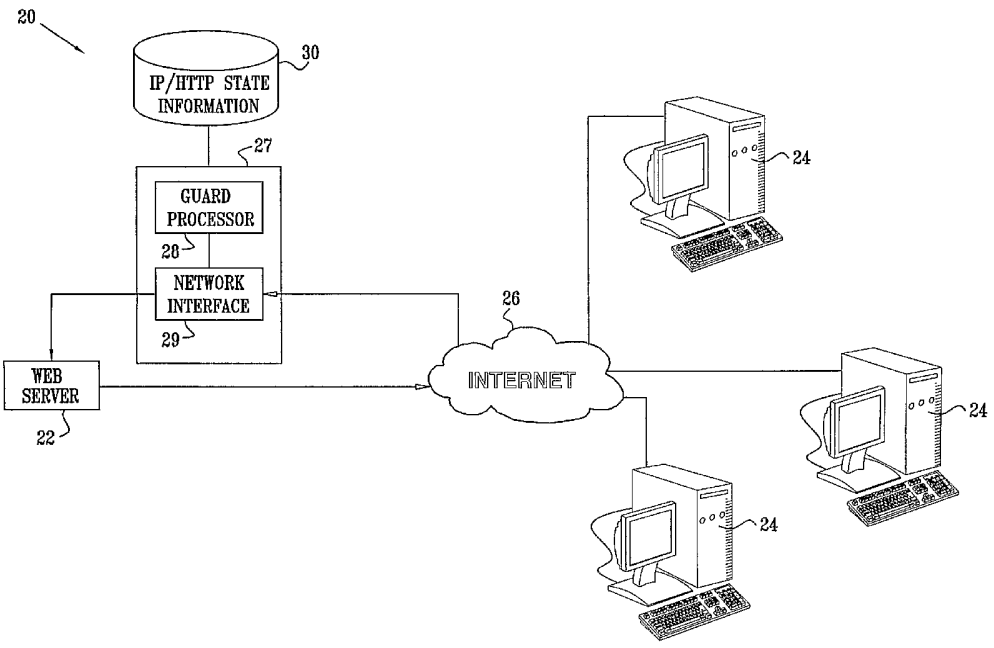
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

[Continued on next page]

(54) Title: UPPER-LEVEL PROTOCOL AUTHENTICATION



(57) Abstract: A method for authenticating communication traffic includes receiving a first message, sent over a network from a source address, requesting information from a server in accordance with a higher-level protocol. A challenge is sent to the source address in reply to the first message, in accordance with the higher-level protocol. A second message is received from the source address following the challenge, and the legitimacy of the source address is assessed by determining whether the second message contains a correct response to the challenge.

WO 2005/069732 A2

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

UPPER-LEVEL PROTOCOL AUTHENTICATION

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims the benefit of U.S. Provisional Patent Application 60/539,327, filed January 26, 2004, and is related to the following co-pending applications: U.S. Patent Application 09/929,877, filed August 14, 2001; U.S. Patent Application 10/232,993, filed August 29, 2002; U.S. Patent Application 10/251,912, filed September 20, 2002; and PCT Patent Application PCT/IL02/00996, filed December 10, 2002. All of these related applications are assigned to the assignee of the present patent application and are incorporated herein by reference.

FIELD OF THE INVENTION

The present invention relates generally to computer networks, and specifically to methods and systems for protecting against denial of service and worm attacks in computer networks.

BACKGROUND OF THE INVENTION

In a Denial-of-Service (DoS) attack, an attacker bombards a victim network or server with a large volume of message traffic. The traffic overload consumes the victim's available bandwidth, CPU capacity, or other critical system resources, and eventually brings the victim to a situation in which it is unable to serve its legitimate clients. Distributed DoS (DDoS) attacks can be even more damaging, as they involve creating artificial network traffic from multiple sources simultaneously.

In order to launch an effective DDoS attack, an attacker typically attempts to control a large number of servers on the Internet. One approach to gaining such

control is to use "worms," which are malicious programs that self-replicate across the Internet by exploiting security flaws in widely-used services. Worm infections are often invisible to the user of an infected computer, and the worm may copy itself to other computers independently of any action taken by the computer user. After taking control of a computer, the worm often uses the computer to participate in a DDoS attack, without any knowing collaboration on the part of the computer user. Infected computers that participate in this sort of mass malicious activity are referred to herein as "zombies."

The present invention will be more fully understood from the following detailed description of the embodiments thereof, taken together with the drawings in which:

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram that schematically illustrates a computer network system, in accordance with an embodiment of the present invention;

Fig. 2 is a flow chart that schematically illustrates a method for protecting against DDoS attacks, in accordance with an embodiment of the present invention; and

Fig. 3 is a message flow diagram that schematically shows details of a method for authenticating a source of incoming packet traffic, in accordance with an embodiment of the present invention.

DETAILED DESCRIPTION OF EMBODIMENTS

Many DDoS attacks use "spoofed" IP packets - packets containing a bogus IP source address - making it difficult for the victim network to identify the source of the attack. In response to this problem, the above-mentioned related applications describe methods that may

be used to determine whether the IP source address of an incoming packet is authentic or spoofed. Traffic from authentic IP addresses may then be passed on to its intended destination, while packets with spoofed addresses are blocked.

Zombies, however, may have legitimate IP addresses (belonging to the infected source computer), and anti-spoofing measures may therefore fail to filter out the packets generated by such zombies during a DDoS attack. Thus, in a typical attack, many zombies may succeed in establishing TCP connections with a victim server, and then may use these connections to bombard the server with messages, such as HTTP requests.

Embodiments of the present invention provide methods for resisting this sort of attack, by distinguishing legitimate messages from messages sent by zombies. For this purpose, some embodiments of the present invention enable a network guard device to challenge sources of incoming packet traffic so as to determine whether the sources comply fully with higher-level communication protocols, such as HTTP (including features of HTML) or DNS, which operate above the transport layer (typically TCP or UDP). Failure of a computer at a given source IP address to comply with the higher-level protocol indicates that the source may be a zombie, and incoming packets from this source are therefore blocked.

Fig. 1 is a block diagram that schematically illustrates a computer network system 20, in accordance with a preferred embodiment of the present invention. A Web server 22 communicates with clients 24 via a wide-area network (WAN) 26, typically the Internet. To prevent DDoS attacks on server 22, a guard device 27 intercepts incoming HTTP request packets from network 26 that are addressed to server 22. Guard device 27

comprises a guard processor 28, which performs the various protection and authentication methods described herein, and a network interface 29, which communicates with other components of system 20 and with WAN 26. The guard processor checks the IP source address of each packet that it intercepts against reference values stored in a database 30 or other data structure. Methods for generating these reference values - indicating which requests are legitimate, and which may have originated from spoofed IP addresses or from zombies - are described further hereinbelow. The guard processor blocks illegitimate requests from passing through to server 22.

The configuration and operation of guard device 27 are shown and described herein by way of example, and alternative configurations and modes of operation will be apparent to those skilled in the art. For example, rather than being connected in-line with server 22, as shown in Fig. 1, guard device 27 may be connected in other configurations, for example, by a "lollipop" connection to a router (not shown) that forwards packets to server 22. Alternatively, functions of the guard device may be integrated into the router or server or into other network equipment, such as a firewall. These and other possible operational configurations of the guard device are described in the above-mentioned related applications. Note that although guard device 27 is shown and described herein as protecting a single server 22, in practice one or more guard devices of this sort may be deployed to protect a group of computers, such as a cluster of servers or an entire LAN. Additional deployment scenarios for the guard device(s) (not necessarily zombie-based) are described in the above-mentioned related applications.

Typically, guard device 27 comprises a general-purpose computer, which is programmed in software to carry out the functions described herein. The software may be downloaded to the computer in electronic form, over a network, for example, or it may alternatively be supplied to the computer on tangible media, such as CD-ROM. Further alternatively, guard device 27 may be implemented in dedicated hardware logic, or using a combination of hardware and software elements.

Fig. 2 is a flow chart that schematically illustrates a method that is carried out by guard processor 28 for protection against DDoS attacks, in accordance with an embodiment of the present invention. The guard processor may perform its packet screening and verification functions at all times, or it may alternatively filter packets only under stress conditions, in which a DDoS attack on server 22 is suspected. In order to determine whether an attack may be in progress, guard processor 28 intercepts incoming traffic via network interface 29 and monitors selected statistical characteristics of the incoming traffic that is directed to server 22, at an attack detection step 50. For example, the guard processor may use one or more of the following criteria to detect a zombie-based DDoS attack:

- Number and distribution of source IP addresses - A sudden change, such as an increase in the number of different source IP addresses attempting to communicate with the server, may be indicative of an attack.
- Distribution of user agents specified in HTTP requests - The agent field in the HTTP request is optional, but it is usually used to specify the type

of browser submitting the request. A sudden change in the distribution of agents may indicate that a large fraction of the requests are being submitted by zombies, which specify a particular agent as dictated by the malicious program that is controlling them. (In order to limit the amount of malicious traffic that can reach server 22, guard processor 28 may optionally determine, in the absence of an attack, a baseline percentage distribution of HTTP requests among the different possible user agents, and may then simply block all traffic specifying a particular user agent that is in excess of the baseline percentage for that agent.) Similarly a sudden change in the number of requests without the agent field, or with a bogus agent field, may be indicative of a zombie attack.

- Occurrence of other regular patterns in the incoming traffic - Zombies tend to send many identical packets repeatedly, at regular intervals. Detection of this sort of repeating pattern may be indicative of an attack.

Other attack detection criteria will be apparent to those skilled in the art. Additional criteria (not necessarily zombie-based) are described in the above-mentioned related applications.

As long as no attack in progress, guard processor 28 typically permits incoming packets to pass through to server 22, at a packet delivery step 52. On the other hand, when an attack is believed to be in progress, guard processor 28 filters some or all of the incoming traffic, at a filtering step 54. For this purpose, the guard processor maintains a record in database 30 of IP source addresses that are known to be legitimate (because of past communications with these source addresses, as

described below). Database 30 may also contain a "blacklist" of addresses that are believed to be malicious. Guard processor 28 checks the source address of each incoming packet against the database record, at a source address checking step 56. If the address appears on the legitimate list, the packet is passed on to server 22 at step 52. (Additionally or alternatively, if the address appears on the blacklist, the packet may be discarded.)

If the IP source address of the incoming packet is unknown, guard processor 28 tests the address to determine whether it is legitimate or spoofed, at a spoofing check step 58. Typically, the guard processor initiates a challenge/response routine, by sending a packet (the "challenge") containing certain information to the IP source address of the incoming packet via interface 29. The guard processor then checks that the response packet received from the IP source address contains appropriate matching information, at an IP authentication step 60. Various challenge/response methods that may be used for this purpose are described in the above-mentioned U.S. Patent Application 10/232,993. If the IP address is found to be bogus, the incoming packet is discarded, and the address may be entered in the blacklist in database 30, at a packet discard step 62.

Fig. 3 is a message flow diagram, which shows details of spoofing check step 58, in accordance with one embodiment of the present invention. In this example, the TCP three-way handshake is used to authenticate the source IP address. The message flow begins when guard processor 28 intercepts a TCP SYN packet sent from an IP source address that does not yet appear in database 30. The SYN packet has a certain packet sequence number (s#),

in accordance with TCP convention. The guard processor sends back a TCP SYN-ACK packet to the IP source address of the SYN packet via interface 29. The SYN-ACK packet contains an encoded cookie (c#), which is encoded in the sequence number (s#) of the packet. Any suitable method of cookie generation that is known in the art may be used for this purpose and for generating cookies in other embodiments of the present invention. In one embodiment, a hash generator implements a hash function for mapping packet attributes, such as the IP source address and port, to cookies. The hash generator calculates a hash value, which is used as a key for indexing a cookie table comprising a set of random cookies. The random cookie values are replaced after use to prevent an attacker who succeeds in discovering a legitimate cookie value from re-using the cookie.

If guard processor 28 then receives a proper TCP ACK packet back from the same IP source address, identified by the proper sequence number and cookie, the guard processor is able to ascertain that the source address is legitimate, rather than spoofed. (Note, however, that the guard processor still does not know whether the computer at this source address is a zombie or not). Alternative anti-spoofing methods are described in the above-mentioned related applications.

Returning now to Fig. 2, after guard processor 28 verifies that the IP source address of a given packet is authentic at step 60, it may go on to test the legitimacy of the higher-level software running on the source computer, at a protocol challenge step 64. In the present embodiment, it is assumed that guard device 27 is protecting a Web server (as shown in Fig. 1), and that the guard processor has intercepted a HTTP request from an unknown source address. Step 64 tests whether the

HTTP request was generated by a legitimate browser, complying with all the requirements of HTTP. Based on this test, the guard processor determines whether the source computer is legitimate or a zombie, at a browser legitimation step 66. An exemplary test of this sort is described below with reference to Fig. 3.

The test used at step 64 is based on sending a HTTP response, containing a HTML directive (the challenge), back to the IP source address of the incoming HTTP request, and checking the next reply returned from this IP address. For the most part, zombies are driven by relatively simple programs, which may be capable of emulating certain basic aspects of HTTP, but do not implement all the specified functions of HTML (as required, for example, by IETF RFC 1866 and the applicable HTML specification, such as HTML 4.0). Therefore, if the source address returns the reply that is expected in compliance with the protocol, guard processor 28 may conclude that the computer at the IP source address is legitimate, and is not a zombie. In this case, guard processor 28 adds the IP address to the list of legitimate addresses in database 30, at an address approval step 68. Packets from this address may now be delivered to server 22 at step 52. Otherwise, if the computer at the IP source address failed to respond to the challenge or responded incorrectly, the incoming packet is discarded at step 62, and its IP source address may be added to the blacklist.

Fig. 3 illustrates one type of test that may be used at step 64. In this example, it is assumed that after the source computer on network 26 establishes its TCP connection with guard device 27 (at step 58), it submits a HTTP request for a certain URI on server 22, for example, GET /index.html. As noted above, the request

may also specify other HTTP fields, such as the user agent. The guard processor intercepts this request via interface 29 and returns a response, which redirects the source computer to refresh its browser with a new URI (identified in Fig. 3 as URI'). Requests directed to the URI' will also be intercepted by the guard processor, but URI' contains information, such as a cookie, that will enable the guard processor to identify the source of the request. For example, the guard processor may return a response containing the HTML directive: <META HTTP-EQUIV="Refresh CONTENT="1;URL=cookie.index.html">, wherein "cookie" is a unique string generated by the guard processor.

Normally, this response should cause the browser on the source computer to open a new TCP connection with guard processor 28, and then resubmit its HTTP request to URI', i.e., to "cookie.index.html". (To open a new TCP connection, the source computer again sends a SYN packet, receives a SYN-ACK from the guard or the target, and then sends an ACK. These three-way exchanges associated with the HTTP GET URI' and the final HTTP GET URI are omitted from Fig. 3 for the sake of simplicity.) Upon receiving this new request, the guard processor is able to conclude that it is communicating with a legitimate browser on the source computer, and adds the IP address of the source computer to its approved list in database 30. The guard processor then redirects the source computer once again to the original URI=index.html. As a result, the source computer will attempt to open yet another TCP connection with server 22. This time, however, the guard processor will recognize the IP source address of the TCP SYN packet from the source computer as legitimate, and will pass the packet through to server 22. The server and

source computer may then proceed to communicate in the normal fashion.

On the other hand, if the original HTTP request from the source computer was sent by a zombie, rather than by a legitimate browser, the source computer will be unable to parse the HTTP response sent back by guard processor 28. Therefore, the source computer will not resubmit its request to "cookie.index.html". Rather, the source computer will, in all likelihood, simply continue submitting further requests to the original URI. Since the guard processor will not have authenticated the IP source address, it will not permit these requests to pass through to server 22. Furthermore, upon receiving multiple, repeated requests of this sort, the guard processor may conclude that the source of the requests is a zombie, and will then add the IP source address to the blacklist.

Various other methods may be used at step 64 in order to verify that a legitimate browser is operating at a given IP source address. These methods may be based on encoding cookies in other parts of the HTTP response sent by guard processor 28, or by testing the source computer for compliance with other aspects of the applicable protocols, such as RFC 1866 or RFC 2616. For example, the guard processor may redirect the browser on the source computer by replying to the initial HTTP request with a HTTP redirect response (status code 307), redirecting the client browser to URI', containing the encoded cookie.

Alternatively or additionally, the response sent by the guard processor may test whether the original HTTP request sent by the source computer was submitted in response to instructions of a human operator of the source computer. For example, the response may cause the

browser on the source computer to display an image or play a sound, and prompt the human operator to type a corresponding word into the computer. The response causes the source computer to return the word that the user has typed, thus permitting the guard processor to verify the presence of a human user operating the browser on the source computer. A zombie, clearly, will fail this test. Challenge/response routines of this sort, for verifying the presence of a human user on the source computer, are described further in the above-mentioned U.S. Patent Application 09/929,877.

Although the embodiment described above makes reference particularly to HTTP and its use in conjunction with Web server 22, the principles of the present invention are generally applicable to authentication of incoming traffic using higher-level protocols of other types. In the context of the present patent application, the term "higher-level protocol" refers to protocols operating above the transport layer (layer 4), as defined by the well-known Open Systems Interconnection (OSI) Reference Model. Internet traffic generally uses TCP or UDP as its transport-layer protocol. Higher-level protocols that may operate over TCP or UDP include (but are not limited to) HTTP, FTP, DNS, RTP, POP/SMTP, SNMP, Usenet, Telnet and NFS. These protocols are generally classified as "presentation-layer" protocols, although this is a loose classification, and these protocols are also often referred to as "application-layer" protocols.

In any case, when clients attempt to communicate with a server according to any higher-level protocol such as these, a guard device protecting the server may use a challenge/response technique based on the requirements of the specific protocol in order to authenticate the sources of the communications. For example, the above-

mentioned U.S. Patent Application 10/251,912 describes methods and devices for distinguishing between spoofed and authentic DNS requests. Many other higher-level protocols (in addition to those listed above) are known in the art, and are amenable to authentication by the methods of the present invention.

Furthermore, although the embodiments described above are directed mainly to processing IP packet traffic sent over the Internet, the principles of the present invention are similarly applicable to networks of other types, using other protocol families. It will thus be appreciated that the embodiments described above are cited by way of example, and that the present invention is not limited to what has been particularly shown and described hereinabove. Rather, the scope of the present invention includes both combinations and subcombinations of the various features described hereinabove, as well as variations and modifications thereof which would occur to persons skilled in the art upon reading the foregoing description and which are not disclosed in the prior art.

CLAIMS

1. A method for authenticating communication traffic, comprising:

receiving a first message, sent over a network from a source address, requesting information from a server in accordance with a higher-level protocol;

in reply to the first message, sending a challenge to the source address in accordance with the higher-level protocol;

receiving a second message from the source address following the challenge; and

assessing legitimacy of the source address by determining whether the second message contains a correct response to the challenge.

2. The method according to claim 1, wherein the correct response is determined by a specification of the higher-level protocol.

3. The method according to claim 1, wherein sending the challenge comprises encoding information in a reply message sent to the source address in reply to the first message, and wherein assessing the legitimacy comprises ascertaining whether the second message contains the encoded information.

4. The method according to claim 3, wherein the higher-level protocol comprises HTTP, and wherein encoding the information comprises sending the reply message so as to redirect a browser at the source address to a URI that contains a cookie.

5. The method according to any of the preceding claims, wherein assessing the legitimacy comprises verifying that the second message was generated by a computer at the

source address responsively to an input made to the computer by a human operator.

6. A method for processing communication traffic, comprising:

monitoring the communication traffic that is sent over a network to a destination address;

determining a baseline characteristic of the communication traffic while a DDoS attack is not in progress;

detecting a deviation from the baseline characteristic that is indicative that at least some of the communication traffic has been sent by zombies having legitimate addresses on the network; and

responsively to detecting the deviation, filtering the communication traffic so as to remove at least some of the communication traffic sent by the zombies.

7. The method according to claim 6, wherein detecting the deviation comprises detecting an increase in a number of different source addresses from which the communication traffic has originated.

8. The method according to claim 6 or 7, wherein monitoring the communication traffic comprises reading contents of incoming data packets sent from the legitimate addresses on the network in accordance with a higher-level protocol, and wherein detecting the deviation comprises detecting a pattern in the contents.

9. Apparatus for authenticating communication traffic, comprising:

a network interface, which is arranged to communicate with a network; and

a processor, which is coupled to the network interface and is arranged to receive a first message sent over the network from a source address, requesting

information from a server in accordance with a higher-level protocol, to send a challenge to the source address in reply to the first message, in accordance with the higher-level protocol, to receive a second message from the source address following the challenge, and to assess legitimacy of the source address by determining whether the second message contains a correct response to the challenge.

10. The apparatus according to claim 9, wherein the correct response is determined by a specification of the higher-level protocol.

11. The apparatus according to claim 9, wherein the processor is arranged to encode information in a reply message sent to the source address in reply to the first message and to assess the legitimacy of the source address by ascertaining whether the second message contains the encoded information.

12. The apparatus according to claim 11, wherein the higher-level protocol comprises HTTP, and wherein the processor is arranged to send the reply message so as to redirect a browser at the source address to a URI that contains a cookie.

13. The apparatus according to any of claims 9-12, wherein the processor is arranged to verify that the second message was generated by a computer at the source address responsively to an input made to the computer by a human operator.

14. Apparatus for processing communication traffic, comprising:

a network interface, which is arranged to communicate with a network; and

a processor, which is coupled to the network interface and is arranged to monitor the communication traffic that is sent over the network to a destination address, to determine a baseline characteristic of the communication traffic while a DDoS attack is not in progress, to detect a deviation from the baseline characteristic that is indicative that at least some of the communication traffic has been sent by zombies having legitimate addresses on the network, and to filter the communication traffic responsively to detecting the deviation, so as to remove at least some of the communication traffic sent by the zombies.

15. The apparatus according to claim 14, wherein the processor is arranged to detect an increase in a number of different source addresses from which the communication traffic has originated.

16. The apparatus according to claim 14 or 15, wherein the processor is arranged to read contents of incoming data packets sent from the legitimate addresses on the network in accordance with a higher-level protocol and to detect a pattern in the contents.

17. A computer software product for authenticating communication traffic, the product comprising a computer-readable medium in which program instructions are stored, which instructions, when executed by one or more processors, cause the one or more processors to receive a first message sent over a network from a source address, requesting information from a server in accordance with a higher-level protocol, to send a challenge to the source address in reply to the first message, in accordance with the higher-level protocol, to receive a second message from the source address following the challenge, and to assess legitimacy of the source address by determining

whether the second message contains a correct response to the challenge.

18. The product according to claim 17, wherein the correct response is determined by a specification of the higher-level protocol.

19. The product according to claim 17, wherein the instructions cause the one or more processors to encode information in a reply message sent to the source address in reply to the first message and to assess the legitimacy of the source address by ascertaining whether the second message contains the encoded information.

20. The product according to claim 19, wherein the higher-level protocol comprises HTTP, and wherein the instructions cause the one or more processors to send the reply message so as to redirect a browser at the source address to a URI that contains a cookie.

21. The product according to any of claims 17-20, wherein the instructions cause the one or more processors to verify that the second message was generated by a computer at the source address responsively to an input made to the computer by a human operator.

22. A computer software product for processing communication traffic, the product comprising a computer-readable medium in which program instructions are stored, which instructions, when executed by one or more processors, cause the one or more processors to monitor the communication traffic that is sent over a network to a destination address, to determine a baseline characteristic of the communication traffic while a DDoS attack is not in progress, to detect a deviation from the baseline characteristic that is indicative that at least some of the communication traffic has been sent by

zombies having legitimate addresses on the network, and to filter the communication traffic responsively to detecting the deviation, so as to remove at least some of the communication traffic sent by the zombies.

23. The product according to claim 22, wherein the instructions cause the one or more processors to detect an increase in a number of different source addresses from which the communication traffic has originated.

24. The product according to claim 22 or 23, wherein the instructions cause the one or more processors to read contents of incoming data packets sent from the legitimate addresses on the network in accordance with a higher-level protocol and to detect a pattern in the contents.

25. Apparatus for authenticating communication traffic, comprising:

means for receiving a first message sent over a network from a source address, requesting information from a server in accordance with a higher-level protocol;

means for sending a challenge to the source address in reply to the first message, in accordance with the higher-level protocol;

means for receiving a second message from the source address following the challenge; and

means for assessing legitimacy of the source address by determining whether the second message contains a correct response to the challenge.

26. Apparatus for processing communication traffic, comprising:

means for monitoring the communication traffic that is sent over a network to a destination address;

means for determining a baseline characteristic of the communication traffic while a DDoS attack is not in progress;

means for detecting a deviation from the baseline characteristic that is indicative that at least some of the communication traffic has been sent by zombies having legitimate addresses on the network; and

means for filtering the communication traffic responsively to detecting the deviation, so as to remove at least some of the communication traffic sent by the zombies.

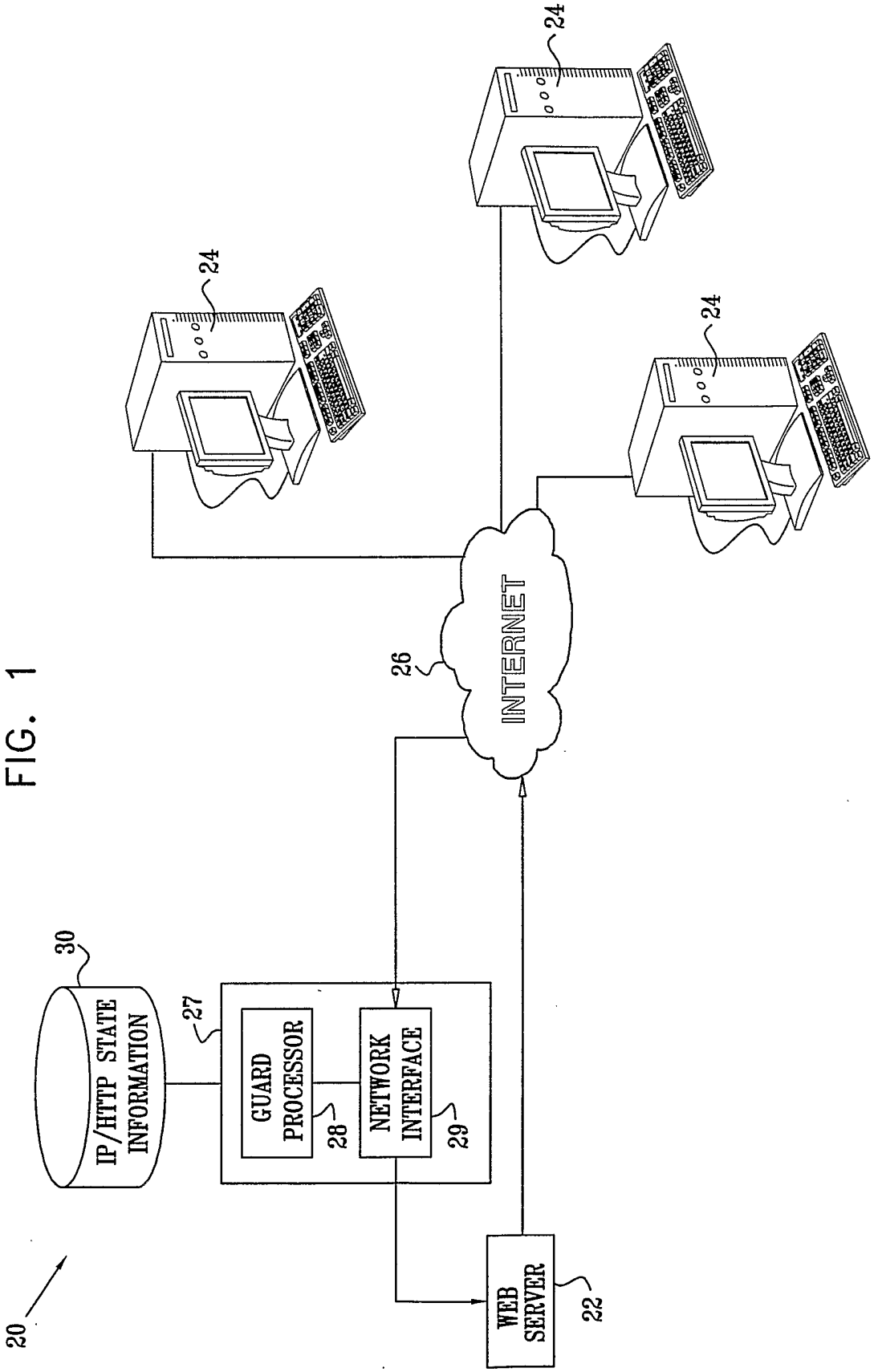
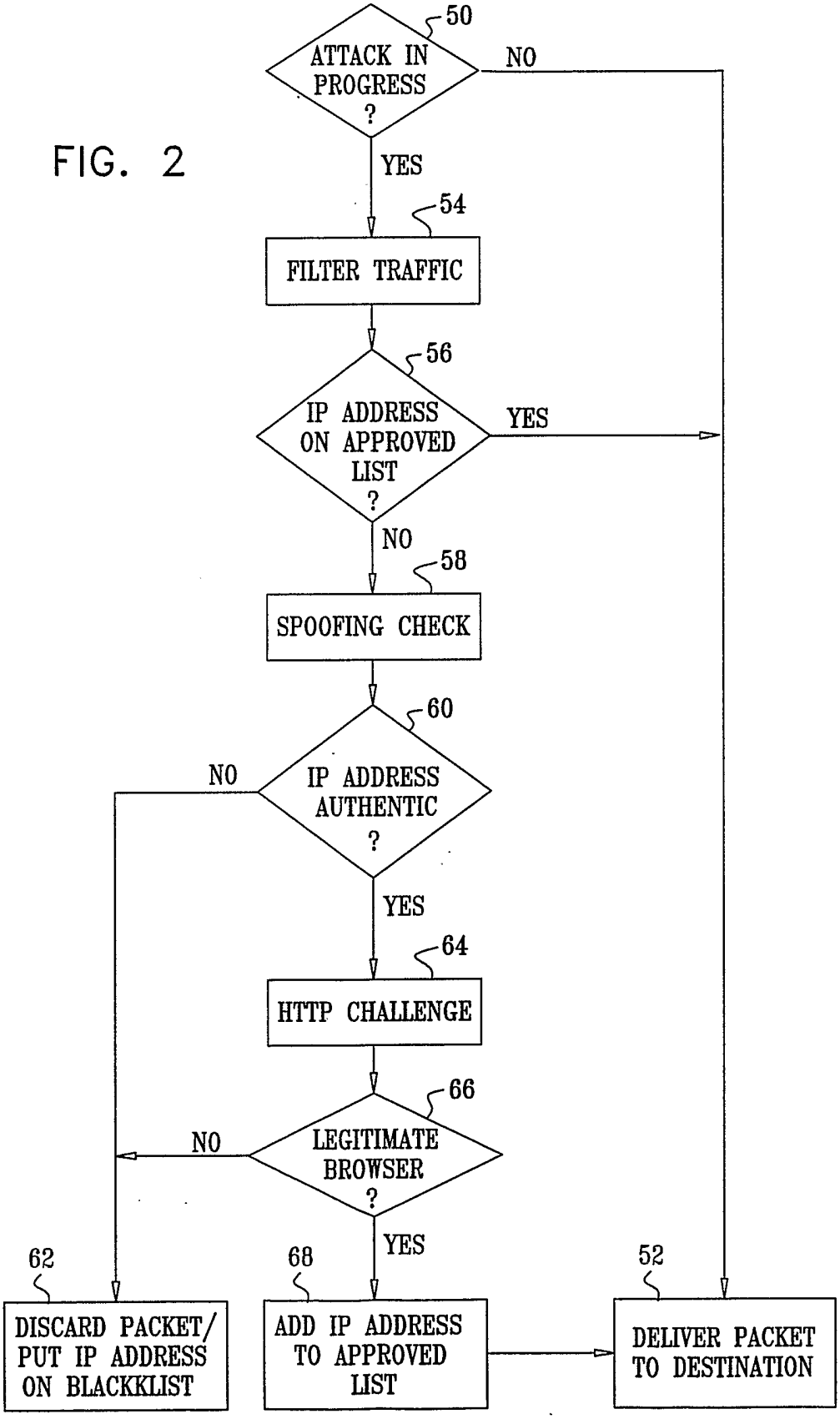

24
24
24
INTERNET
26
30
IP/HTTP STATE INFORMATION
27
GUARD PROCESSOR
28
NETWORK INTERFACE
29
WEB SERVER
22
20

FIG. 1

FIG. 2


50
ATTACK IN PROGRESS ?
NO
YES
54
FILTER TRAFFIC
56
IP ADDRESS ON APPROVED LIST ?
YES
NO
58
SPOOFING CHECK
60
IP ADDRESS AUTHENTIC ?
NO
YES
64
HTTP CHALLENGE
66
LEGITIMATE BROWSER ?
NO
YES
62
DISCARD PACKET/ PUT IP ADDRESS ON BLACKKLIST
68
ADD IP ADDRESS TO APPROVED LIST
52
DELIVER PACKET TO DESTINATION

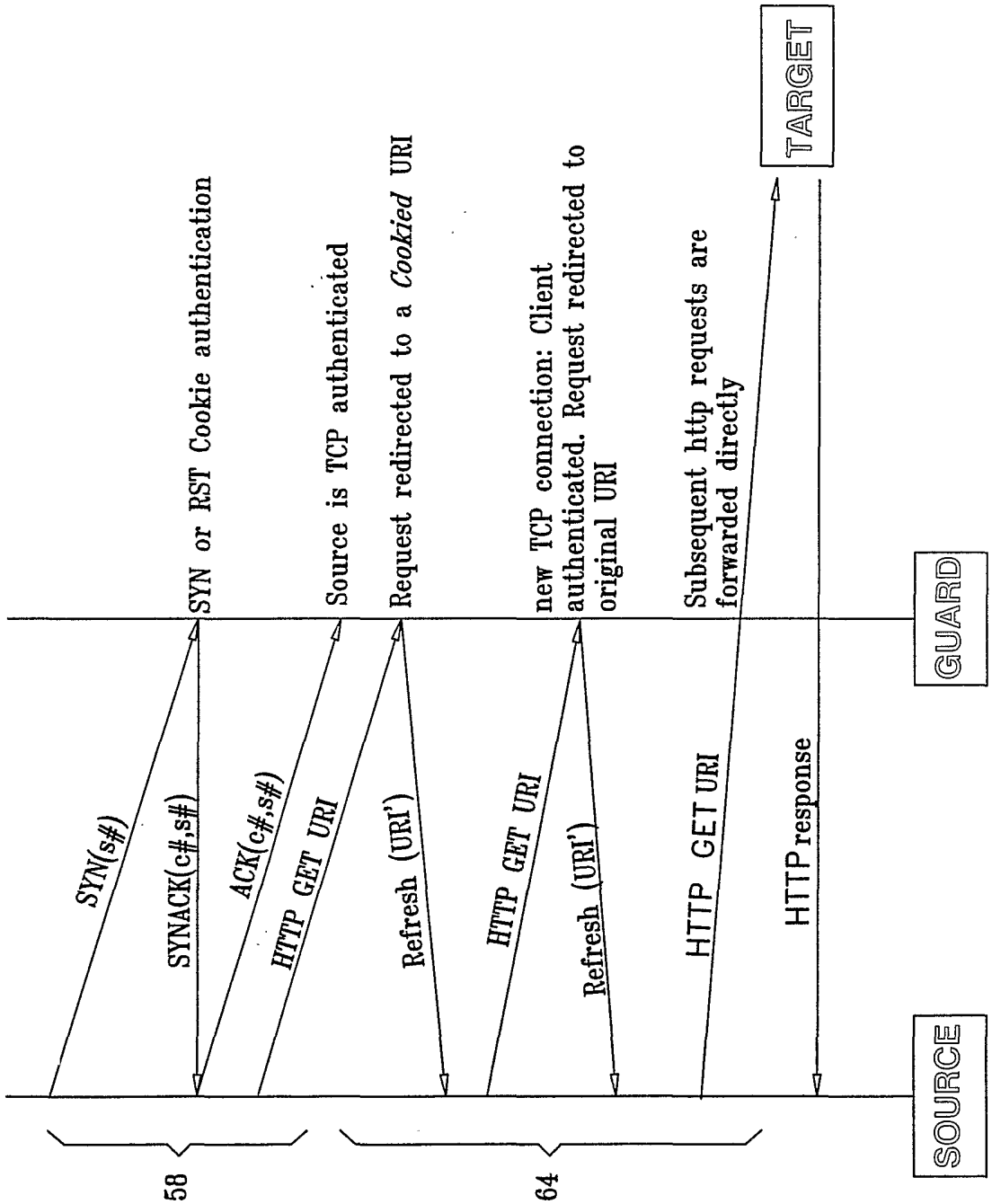

SOURCE
GUARD
TARGET
SYN(s#)
SYNACK(c#,s#)
ACK(c#,s#)
HTTP GET URI
Refresh (URI')
HTTP GET URI
Refresh (URI')
HTTP GET URI
HTTP response
SYN or RST Cookie authentication
Source is TCP authenticated
Request redirected to a Cookied URI
new TCP connection: Client authenticated. Request redirected to original URI
Subsequent http requests are forwarded directly
58
64

FIG. 3