US 20020116643A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2002/0116643 A1

Raanan et al. (43) Pub. Date: Aug. 22, 2002

(54) METHOD AND SYSTEM FOR EXTRACTING APPLICATION PROTOCOL CHARACTERISTICS

(76) Inventors: **Gil Raanan**, Zoran (IL); **Tal Moran**, Tel-Aviv (IL); **Yaroa Galant**, Mountainview, CA (US); **Yuval El-Hanani**, Tel-Aviv (IL); **Eran Reshef**, Sunnyvale, CA (US)

Correspondence Address:
BROWN, RAYSMAN, MILLSTEIN, FELDER & STEINER LLP
900 THIRD AVENUE
NEW YORK, NY 10022 (US)

(21) Appl. No.: **09/974,214**

(22) Filed: **Sep. 14, 2001**

Related U.S. Application Data

(63) Continuation of application No. 09/345,920, filed on Jul. 1, 1999, now Pat. No. 6,311,278, which is a continuation-in-part of application No. 09/149,911, filed on Sep. 9, 1998, now Pat. No. 6,321,337.

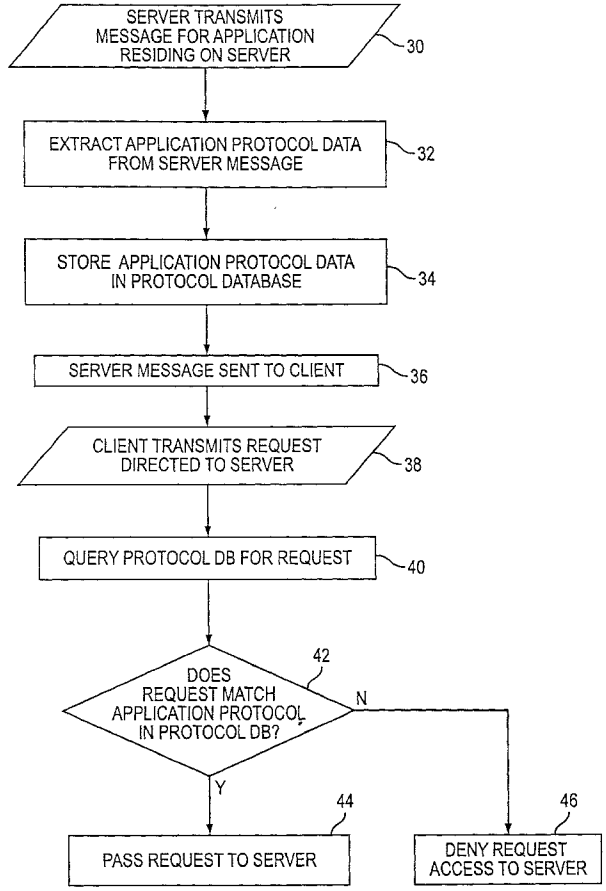
Publication Classification

(51) **Int. Cl.**[7] **G06F 11/30**

(52) **U.S. Cl.** **713/201**

(57) **ABSTRACT**

A method and computer program for automatically and continually extracting application protocols (i.e., defining a set of allowable or authorized actions) for any application. The method involves receiving a message from a server before it is sent or in parallel with sending to a client. The message may be in response to a specific request for it from the client. The program then extracts the application protocol data from the server message. Working with a copy of the message, the program strips off the communications protocol(s) from the message and parses the remaining message to identify user-selectable options contained in the message such as commands, fields, etc. These items represent the set of allowable or authorized user actions for the particular "stage" of the current version of the application as set forth in the message. The set of allowable user actions is then stored by the extraction program in a protocol database accessible to a gateway or filter module.



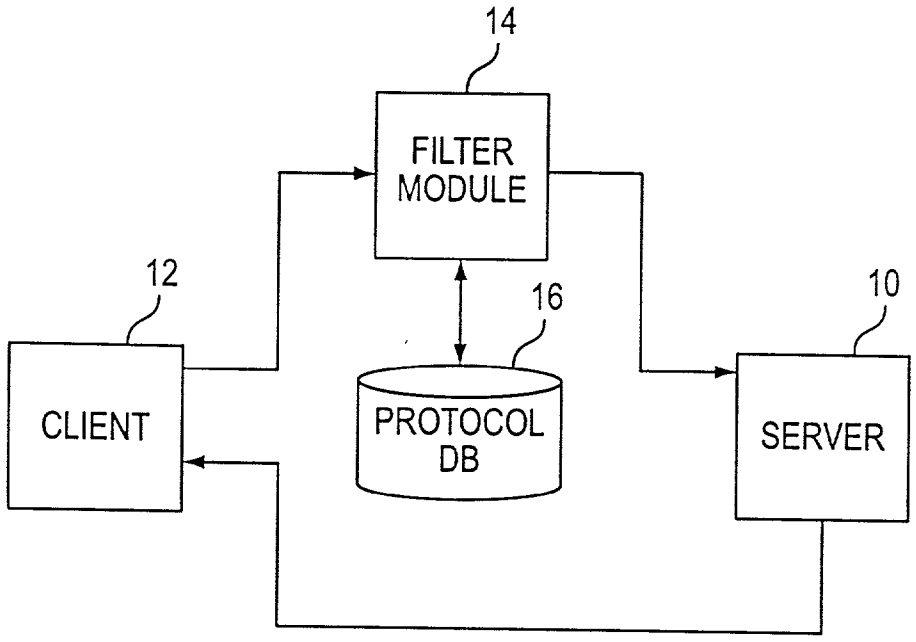

14
FILTER MODULE
12
16
10
CLIENT
PROTOCOL DB
SERVER

FIG. 1

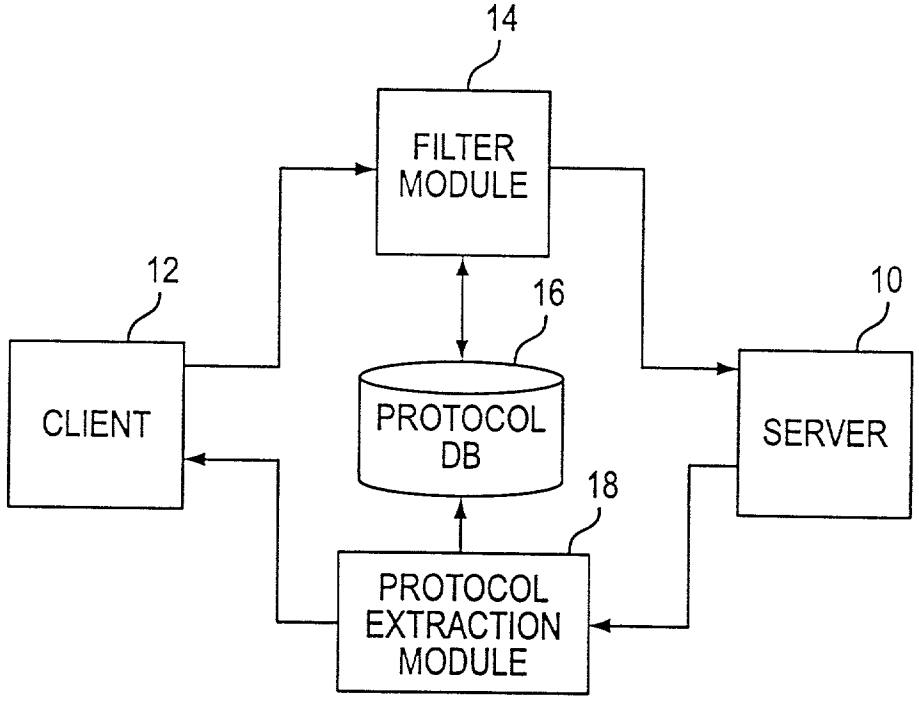

14
FILTER MODULE
12
16
10
CLIENT
PROTOCOL DB
SERVER
18
PROTOCOL EXTRACTION MODULE

FIG. 2

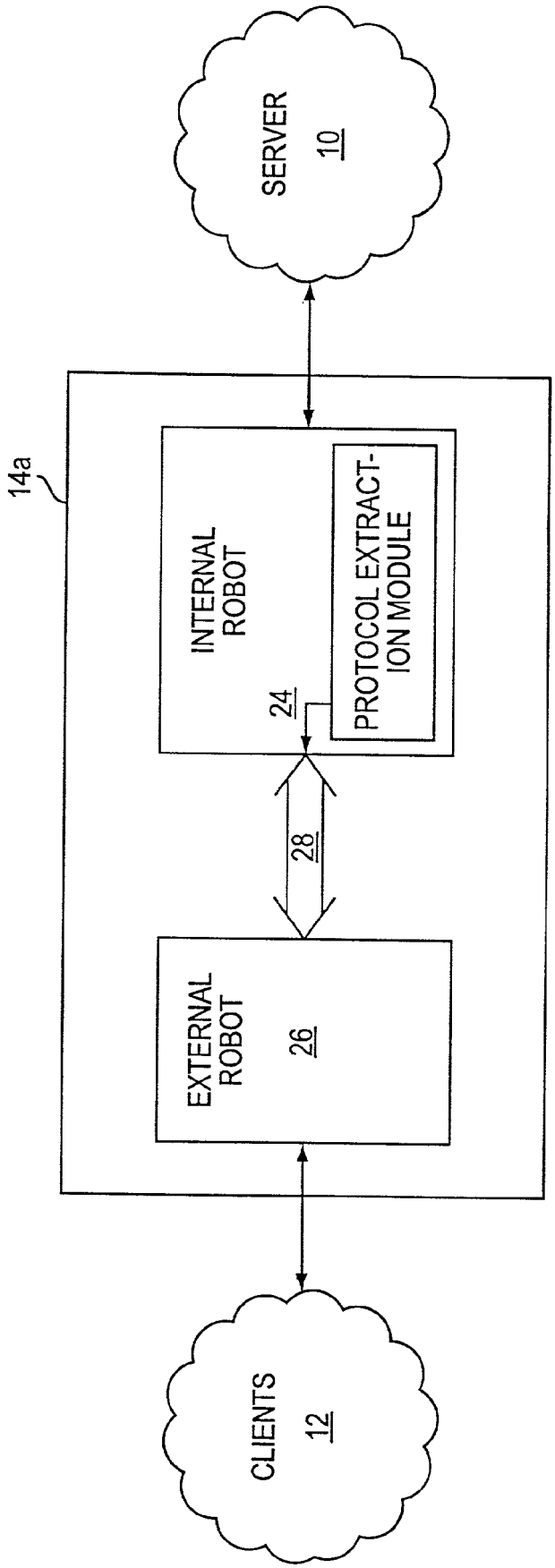

SERVER
10
14a
INTERNAL ROBOT
PROTOCOL EXTRACT-ION MODULE
24
28
EXTERNAL ROBOT
26
CLIENTS
12

FIG. 2A

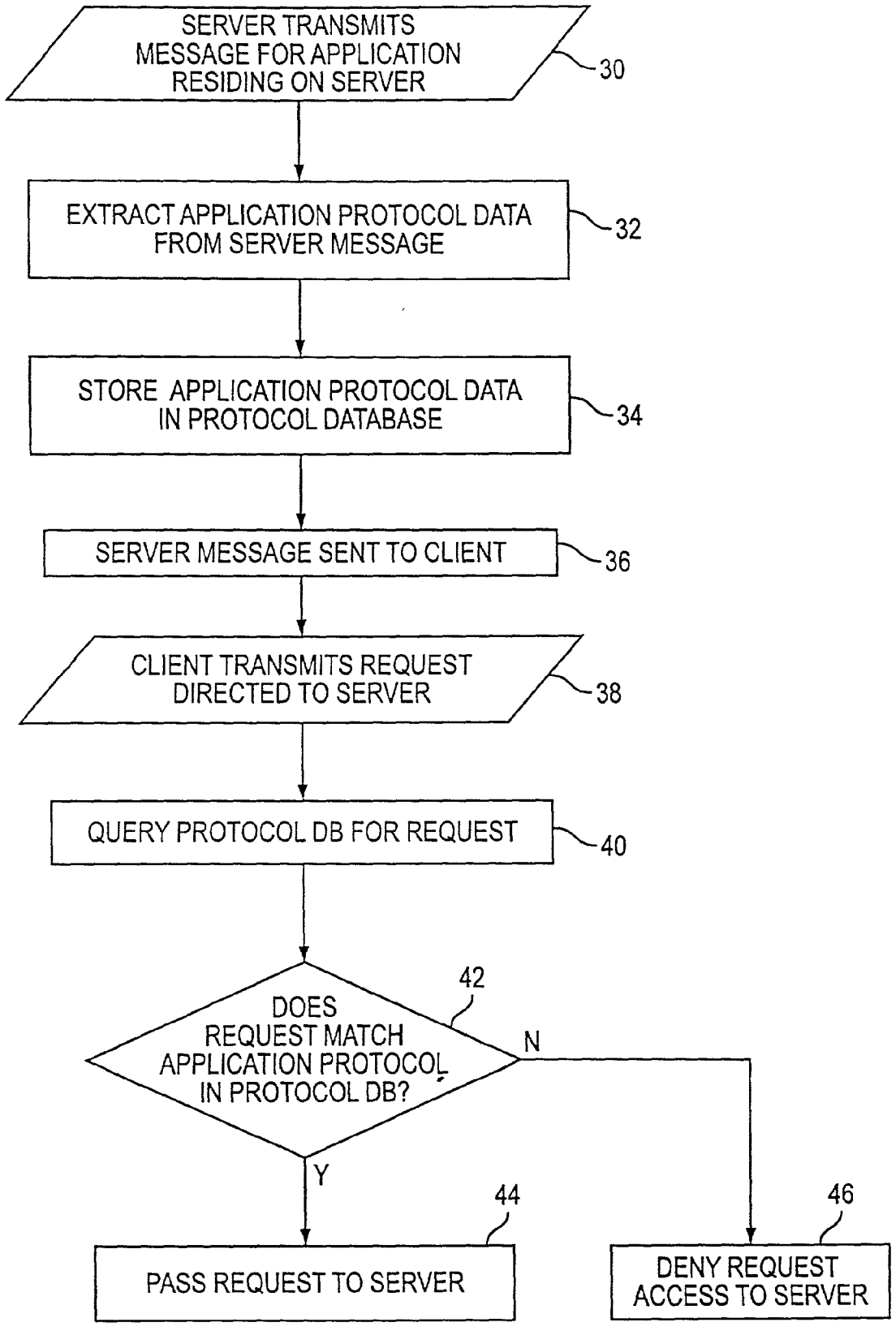

SERVER TRANSMITS MESSAGE FOR APPLICATION RESIDING ON SERVER
30
EXTRACT APPLICATION PROTOCOL DATA FROM SERVER MESSAGE
32
STORE APPLICATION PROTOCOL DATA IN PROTOCOL DATABASE
34
SERVER MESSAGE SENT TO CLIENT
36
CLIENT TRANSMITS REQUEST DIRECTED TO SERVER
38
QUERY PROTOCOL DB FOR REQUEST
40
42
DOES REQUEST MATCH APPLICATION PROTOCOL IN PROTOCOL DB?
N
Y
44
PASS REQUEST TO SERVER
46
DENY REQUEST ACCESS TO SERVER

FIG. 3

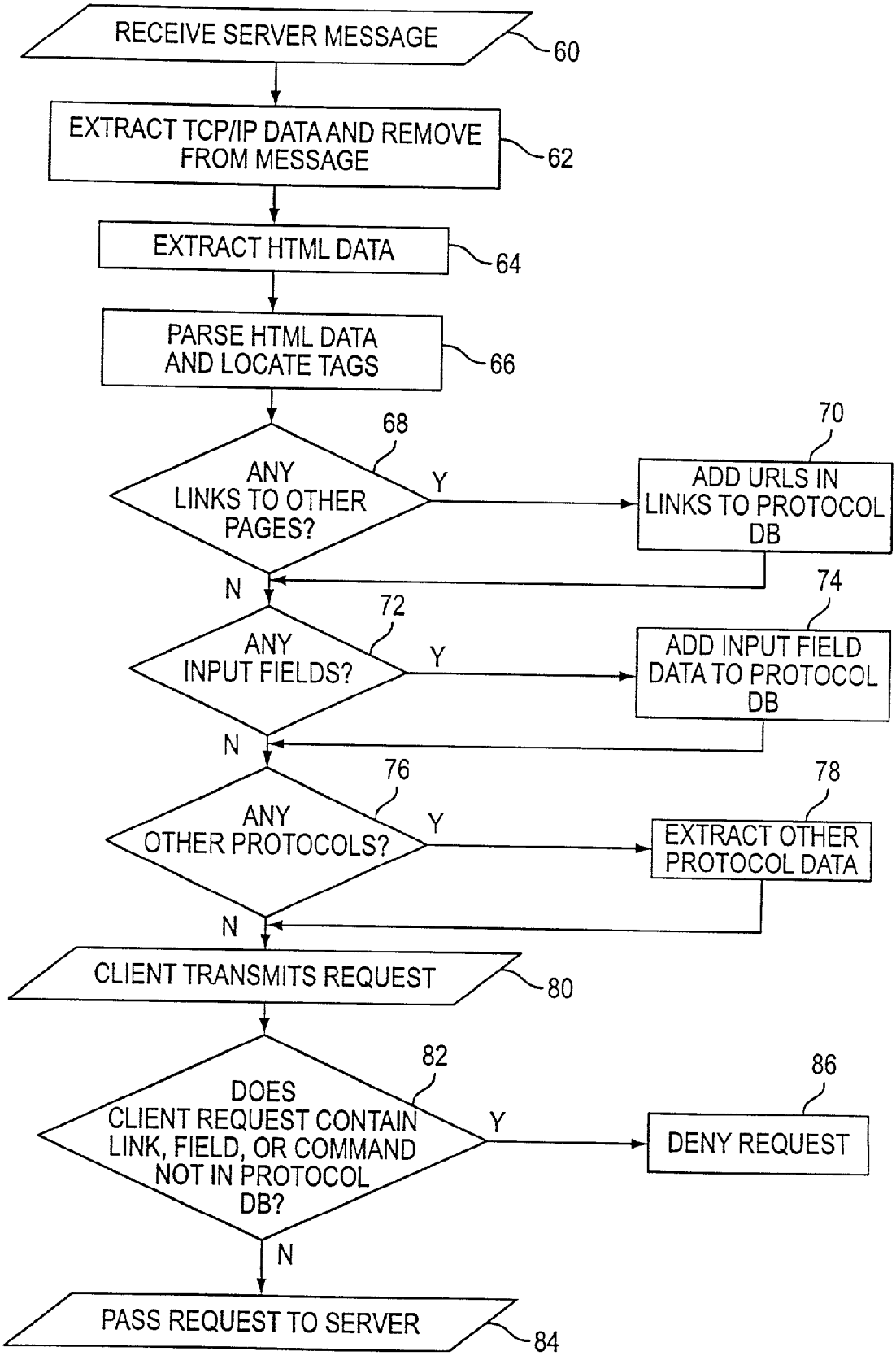

RECEIVE SERVER MESSAGE
60
EXTRACT TCP/IP DATA AND REMOVE FROM MESSAGE
62
EXTRACT HTML DATA
64
PARSE HTML DATA AND LOCATE TAGS
66
68
ANY LINKS TO OTHER PAGES?
Y
70
ADD URLS IN LINKS TO PROTOCOL DB
N
72
ANY INPUT FIELDS?
Y
74
ADD INPUT FIELD DATA TO PROTOCOL DB
N
76
ANY OTHER PROTOCOLS?
Y
78
EXTRACT OTHER PROTOCOL DATA
N
CLIENT TRANSMITS REQUEST
80
82
DOES CLIENT REQUEST CONTAIN LINK, FIELD, OR COMMAND NOT IN PROTOCOL DB?
Y
86
DENY REQUEST
N
PASS REQUEST TO SERVER
84

FIG. 4

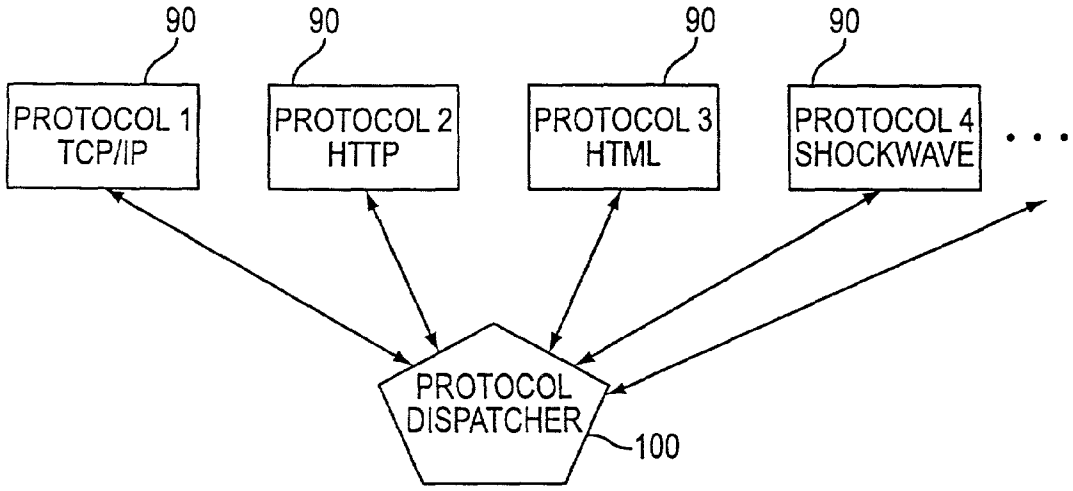

90
90
90
90
PROTOCOL 1
TCP/IP
PROTOCOL 2
HTTP
PROTOCOL 3
HTML
PROTOCOL 4
SHOCKWAVE
PROTOCOL
DISPATCHER
100

FIG. 5

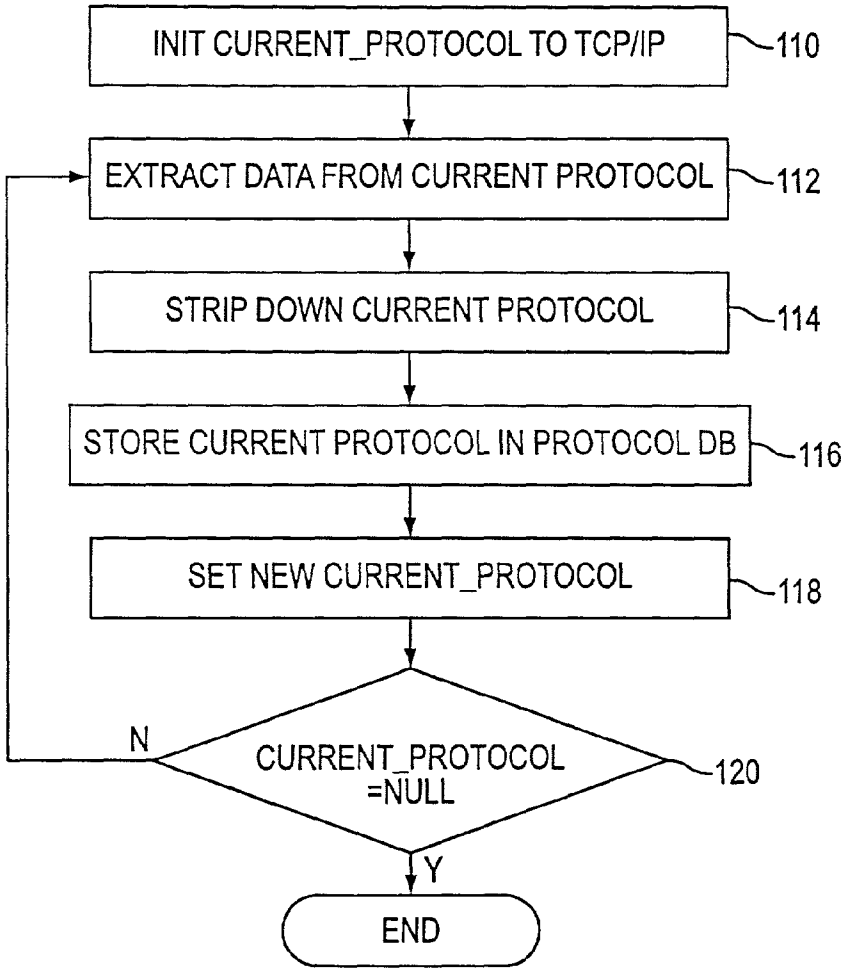

INIT CURRENT_PROTOCOL TO TCP/IP
110
EXTRACT DATA FROM CURRENT PROTOCOL
112
STRIP DOWN CURRENT PROTOCOL
114
STORE CURRENT PROTOCOL IN PROTOCOL DB
116
SET NEW CURRENT_PROTOCOL
118
N
CURRENT_PROTOCOL
=NULL
120
Y
END

FIG. 6

METHOD AND SYSTEM FOR EXTRACTING APPLICATION PROTOCOL CHARACTERISTICS

[0001] This application is a continuation of U.S. application Ser. No. 09/345,920, filed Jul. 1, 1999, now pending, which is a continuation-in-part of U.S. application Ser. No. 09/149,911, filed Sep. 9, 1998, now pending.

COPYRIGHT NOTICE

[0002] A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the Patent and Trademark Office patent files or records, but otherwise reserves all copyright rights whatsoever.

RELATED APPLICATIONS

[0003] This application is related to pending application Ser. No. 09/149,911 titled METHOD AND SYSTEM FOR PROTECTING OPERATIONS OF TRUSTED INTERNAL NETWORKS, filed Sep. 9, 1998, and application Ser. No. 09/150,112 titled METHOD AND SYSTEM FOR MAINTAINING RESTRICTED OPERATING ENVIRONMENTS FOR APPLICATION PROGRAMS OR OPERATING SYSTEMS, filed Sep. 9, 1998, both of which applications are hereby incorporated by reference into this application.

BACKGROUND OF THE INVENTION

[0004] The present invention relates generally to network security and privacy systems, and more particularly to a method and system for continuously and automatically or semi-automatically defining and updating actions which may be taken in an application program operating on a server.

[0005] One way in which the security or privacy of data or an application program residing on a server computer may be compromised is through an unauthorized command. That is, a client computer connectable to the server, such as via the Internet, may transmit a request for the retrieval of data or for the execution of an instruction to which the client is not entitled. For example, a web server accessible over the Internet on which goods are available for sale may allow actions such as the selection of an item to purchase, the input of personal and payment data, or even the execution of an application program to retrieve data previously entered. However, the web server should not allow a given client to change price data or retrieve other data intended to be kept private, and these types of requests should be considered unauthorized or disallowable for that client. Currently, many applications do not include safeguards against clients making these kinds of requests.

[0006] Presently, service provider networks (e.g., commercial sites, government institutes, e-commerce sites, etc.) are often protected by firewall security devices or routers. These tools provide a good level of security against attacks based on the weaknesses of low level protocols (such as TCP or UDP) and of generic Internet applications like FTP or TELNET. However, these tools cannot guard implementations of specific application protocols, such as a specific banking application, billing application, insurance application, etc., nor can they account for changes or updates to application protocols.

[0007] To prevent clients from performing disallowable actions, a gateway or filter mechanism may be interposed between the client and server to identify and eliminate disallowable requests. As shown in **FIG. 1, a** filter module **14** is positioned between a server **10** and clients, only one of which is shown in **FIG. 1** as client **12**. The filter module **14** receives requests from the client **12**, eliminates any disallowable actions requested by the client **12** to the server **10**, and passes the remaining, allowable parts of requests to the server **10**. The filter module **14** determines which requests are allowable by querying a protocol database **16**. The protocol database **16** stores an application protocol for the application program residing on the server. As used herein, an application protocol represents some or all of the allowable actions for the application program.

[0008] An example of a gateway system and related components is described in the aforementioned applications, Ser. Nos. 09/149,911 and 09/150,112, which are incorporated by reference into this application.

[0009] In order to create the protocol database **16**, a developer must know all the protocols of the application and the authorized or allowable actions. However, for applications which utilize complex protocols, the process of specifying the precise protocol can be long and tedious. In addition, the application developer is often not even aware of the complete protocol specification, as implicit assumptions made by the programmer are usually extremely difficult to identify. Furthermore, the developer must monitor changes in the application protocol and update the protocol database accordingly. Failure to have a complete and accurate protocol database could prevent clients from making full use of the application program residing on the server. An ineffective database could alternatively allow clients to take actions which are disallowed in the current version of the application program.

[0010] There is therefore a need for a method and system for at least semi-automatically defining application protocols for applications residing on servers on an on-line, real-time basis.

BRIEF SUMMARY OF THE INVENTION

[0011] It is an object of the present invention to solve the problems described above with security and privacy systems.

[0012] It is another object of the present invention to define allowable actions which may be requested by clients of servers.

[0013] It is another object of the present invention to provide a mechanism for extracting application protocols on an on-line, real-time basis.

[0014] These and other objects are achieved by a method implemented by an extraction computer program for extracting application protocols thereby defining a set of allowable or authorized actions. The method involves receiving a message from a server before it is sent or in parallel with sending to a client. The message may be in response to a specific request for it from the client. In the case of the world wide web, for example, in which clients typically request web documents or pages through browser programs, the requested web page would be intercepted before or in parallel with transmission to the client.

[0015] The extraction program then extracts the application protocol data from the server message. The server message typically contains data for one or more communication protocols required for transmission to the client, such as TCP/IP in the case of Internet communications. Working with a copy of the message, the program parses the communications data from the message and saves or discards this information. Then, the program strips off the communications protocol(s) from the message. The program next parses the remaining message to identify commands, fields, or other user-selectable options contained in the message. These items represent the set of allowable or authorized user actions for the application as set forth in the message.

[0016] The set of allowable user actions is then stored by the extraction program in a protocol database accessible to a gateway or filter module. The protocol data may be stored on a session by session basis, in which case it is used by the filter module to enforce a protocol policy for each individual client/server session and even for each portion or segment of an application program. When used this way, the protocol data may be continuously updated and changed to represent actions which are allowable at any given point. Alternatively, the protocol data may be collected from many sessions over a period of time and stored to create a larger and more complex protocol database.

[0017] In any event, the ability to capture an application protocol from a server message provides for a protocol database which may be continuously updated on an ongoing, real-time basis and which more accurately reflects the set of allowable actions.

BRIEF DESCRIPTION OF THE DRAWINGS

[0018] **FIG. 1** is a block diagram of a client server system having a gateway for filtering client requests;

[0019] **FIG. 2** is a block diagram of the system of **FIG. 1** with the addition of a protocol extraction module in accordance with the present invention;

[0020] **FIG. 2A** is a block diagram of one embodiment of the system of **FIG. 2** in which the gateway comprises external and internal robots;

[0021] **FIG. 3** is a flow chart showing a process of defining allowable actions in an application protocol on an online basis in accordance with the present invention;

[0022] **FIG. 4** is a flow chart showing part of a process of defining allowable actions in an HTML file transmitted from a web server over the Internet in accordance with embodiments of the present invention;

[0023] **FIG. 5** is a block diagram showing a protocol dispatcher component of the protocol extraction module of **FIG. 2** acting on an HTML file in accordance with embodiments of the present invention; and

[0024] **FIG. 6** is a flow chart showing a protocol extraction process performed by the protocol dispatcher component of **FIG. 5**.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0025] Preferred embodiments of the present invention are now described in detail with reference to the drawings in the figures.

[0026] Referring to **FIG. 2, a** computer network such as the Internet, an intranet or any other private network, connects clients **12** and servers **10**, of which only one of each is shown. Associated with the server **10** is a security gateway system consisting of a filter module **14**, a protocol database **16**, and a protocol extraction module **18**. These modules and database may be stored on the server **10**, on a computer separate from and connectable to the server **10**, or on a number of separate but connectable computers.

[0027] The filter module **14** intercepts messages such as requests from the client **12** and queries the protocol database **16** to determine whether the actions or commands in the request are authorized or allowed for the client **12**. The protocol database **16** contains a list of the allowable actions, either for a given client/server session, for a "stage" or segment of the application program, or as a static list of actions allowable for a given application program.

[0028] In some embodiments, the filter module **14** consists of two or more components as described in application Ser. No. 09/149,911, through which commands and other data in the client communication is converted to a simplified protocol for added security. As shown in **FIG. 2**A, the gateway **14***a* contains two separate and distinct processing entities **24**, **26**, referred to herein as robots, connected via a dedicated, secure communication bus **28**. The internal robot **24** is connected to the server **10**, and the external robot **26** is connected to clients **12** via the Internet or other external computing environment. Each robot is capable of translating or reducing a communication or message received from the respective environment to a simplified message using a simplified protocol format referred to herein as a clear inter-protocol or CIP, transmitting the CIP message to the other robot using the inter-robot bus **28** using an inter-robot transfer protocol or IRP, and translating such CIP messages received from the other robot into messages formatted for the respective environment. Together, these three elements **24, 26, 28** implement the protection provided by the gateway **14***a* for the protected internal server **10**. The robots **24**, **26** are two separate and independent logical processes that execute routines defined by respective security gateway software packages. The robots **24**, **26** may be installed on two separate processing devices or one a single processing device operating the one or both of the robots **24**, **26** in protected mode.

[0029] Each of the robots **24**, **26** contains or has access to a protocol manager (not shown) which reduces a message received by the robot for the respective environment to a CIP message for transmission to the other robot, and which also retranslates a message received from the other robot in CIP format into the protocol for the respective native environment. The protocol manager thus uses a database of CIP codes for this reduction and retranslation. As shown in **FIG. 2**A, the protocol extraction module **18**, which resides in the internal robot **24**, extracts protocols in messages received by the internal robot **24** from the server **10**, extracts the protocols as described herein, and provides the application protocol data to robot **26**.

[0030] In accordance with the invention, the protocol extraction module **18** intercepts server messages and extracts application protocol data for addition to the protocol database **16**. The operation of the extraction module **18** in accordance with one embodiment is described with refer-

ence to **FIG. 3**. The server **10** transmits a message directed to the client, step **30**, the message containing information relating to the application residing and running on the server **10** or a computer connected thereto. The message may be a response to a request previously received from the client. Using a copy of the server message or the message itself, the application protocol data is extracted from the server message, step **32**. As described in more detail below, this extraction process may be performed in a number of ways, including through the use of known techniques to identify a low level or communication protocol, such as TCP/IP, stripping such protocol while retaining required data such as IP source data, and searching the remainder of the message for allowed commands or other authorized user actions.

[**0031**] Once extracted, the application protocol data is stored in the protocol database **16**, step **34**. The protocol data may be added to a permanent file relating to the current version of the application, to a temporary, session-based file used for a particular client/server session only, or to a temporary file used only for a particular server message and then overwritten. All of these options allow for the automatic adaptation to changes in an application and for the continuous modification of the protocol database to account for allowable actions in different segments or stages of an application. These options differ to the extent that protocols from prior messages remain relevant for future messages.

[**0032**] The server message is transmitted to the client, step **36**. The client then transmits a request directed to the server, step **38**. The client's request may be a proper response to the server message or may be an attempt to cause the application to execute an unauthorized command. The filter module **14** intercepts the client request, reads it, and queries the protocol database, step **40**. Depending upon the security and privacy desired, the query may need to identify the client, the server, the particular application and/or the particular session.

[**0033**] The request is compared to the application protocol database to determine whether the request is allowable, step **42**. If the request is allowable, the filter module **14** passes the request along to the server, step **44**. If the request does not match any of the actions in the application protocol in the protocol database **16** and is thus considered disallowable, the request is denied access to the server, step **46**, and the client **12** and/or server **10** may be notified of the attempted unauthorized request.

[**0034**] An embodiment of the protocol extraction method used for web-based communications is shown in **FIG. 4**. The extraction module receives the server message, which is a web document or HTML page, step **60**. The TCP/IP protocol data is extracted from the document, step **62**, and saved to help identify the source IP address to, for example, maintain a session with the client to whom the message was addressed. Other communication data such as HTTP is further stripped from the document until the module reads the HTML data, step **64**.

[**0035**] From this data, the module collects information about the design of the particular application. This is accomplished by parsing the HTML document data and locating all the tags, step **66**. For tags such as anchors which define links to other web documents, step **68**, the link with URL is added to the protocol database, step **70**. This applies, for example, with home pages of a web server containing links to many other pages on the server, or with links embedded within certain types of multimedia files such as those contained in Shockwave, RealAudio or RealVideo files. The extraction module also locates any input fields in the web document, step **72**, which may be positioned, for example, within an HTML form. The identity and nature of the field data for such fields, including the type and length of the field, is then added to the protocol database, step **74**. If no field length is specified, a default field length is used. For example, a "name" field is listed in the protocol database as requiring alphanumeric data of a given length in the client request, a date field is listed as requiring date formatted alphanumeric data, and an "email address" field requires email formatted data, e.g., a@b.c.

[**0036**] Using similar steps, the protocol extraction module will also check for forms, fields, fixed fields, hidden fields, menu options, DOM components, etc. For each of these elements, the protocol database will be updated as to their nature and any limitations thereon. For example, for all hidden fields identified, the database will be updated as to their nature and that the client may not change their content.

[**0037**] The extraction module further identifies any other actions available in the web document, step **76**. These include, for example, a "submit" command on an HTML form, a "search" command, or other application-level protocols. These additional actions within the web document are also extracted and stored in the protocol database, step **78**.

[**0038**] Once the gateway or filter receives a client request, step **80**, it compares each link, data, command, or other action in the request with the corresponding entities now stored in the protocol database, step **82**. If no such disallowed actions are in the request, the request is transmitted to the server, step **84**. Otherwise, any link, data, or command not contained in the protocol database is deleted from the request or, alternatively, the entire request is denied, step **86**.

[**0039**] Referring now to **FIGS. 5 and 6**, in some embodiments the protocol extraction module contains a protocol dispatcher **100** which coordinates and manages the extraction process. As shown in **FIG. 5**, the dispatcher **100** extracts one protocol **90** at a time, starting for example, with TCP/IP, HTTP, HTML, and any other protocols. In the web environment, upon receiving a message the dispatcher **100** initializes a variable current_protocol to TCP/IP, step **110** in **FIG. 6**. Data is then extracted from the current protocol, step **112**, and the current protocol is then stripped out of the message, step **114**. The current protocol is then stored in the protocol database, step **116**, or alternatively, the dispatcher **100** may proceed through all the protocols before updating the protocol database.

[**0040**] The variable current_protocol is then incremented or otherwise set to a new protocol, step **118**. If the current_protocol is now NULL, step **120**, meaning that no additional protocols remain to be extracted, the process is complete. Otherwise, the data if any in the message relating to the new current protocol is extracted, step **112**, and the process repeated until complete.

[**0041**] While the invention has been described and illustrated in connection with preferred embodiments, many variations and modifications as will be evident to those skilled in this art may be made without departing from the

spirit and scope of the invention, and the invention is thus not to be limited to the precise details of methodology or construction set forth above as such variations and modification are intended to be included within the scope of the invention.

What is claimed is:

1. A method for defining a set of allowable actions for an application program residing on a server, the method comprising:

receiving a message transmitted by the server addressed to one or more clients;

extracting application protocol data from the server message to thereby retrieve the set of allowable actions which may be taken in response to the server message;

storing the extracted application protocol data in a protocol database.

* * * * *