

(19) World Intellectual Property
Organization
International Bureau



(43) International Publication Date
9 December 2004 (09.12.2004)

PCT

(10) International Publication Number
WO 2004/107132 A2

- (51) International Patent Classification⁷: **G06F**
- (21) International Application Number:
PCT/US2004/017015
- (22) International Filing Date: 28 May 2004 (28.05.2004)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data:
60/473,931 28 May 2003 (28.05.2003) US
- (71) Applicant (for all designated States except US): **CAY-MAS SYSTEMS, INC.** [US/US]; 1179-A North McDowell Boulevard, Petaluma, CA 94954 (US).
- (72) Inventor; and
- (75) Inventor/Applicant (for US only): **ROSENSTEIN, Adam, H.** [US/US]; P.O. Box 750814, Petaluma, CA 94941 (US).
- (74) Agent: **SINGER, James, M.**; Pepper Hamilton LLP, One Mellon Center, 50th Floor, 500 Grant Street, Pittsburgh, 15219 (US).
- (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, IT, 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).

Declarations under Rule 4.17:

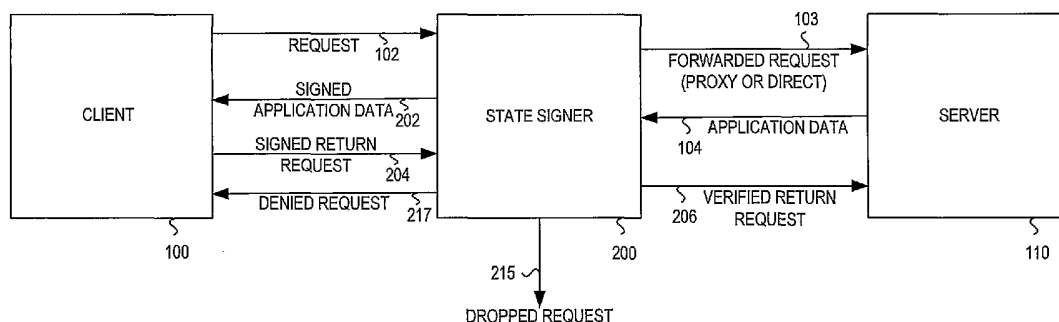
- as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii)) for the following designations AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, 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, UZ, VC, VN, YU, ZA, ZM, ZW, ARIPO patent (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG)
- as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii)) for all designations

Published:

- without international search report and to be republished upon receipt of that report

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.

(54) Title: METHOD, SYSTEM AND SOFTWARE FOR STATE SIGNING OF INTERNET RESOURCES



(57) Abstract: A method, system, and software for state signing of Internet resources is presented in which web pages and other Internet resources are signed after the insertion of metadata indicating intended and authorized uses. In one embodiment, the signing is accomplished through use of a cryptographic signature added to any data item passed to a client that is likely to be passed back to the server later, such as a cookie, URL, or data integrity item. Enabling/disabling of state signing for various data items can be controlled through policies tied to URL prefixes.

METHOD, SYSTEM AND SOFTWARE FOR STATE SIGNING OF INTERNET RESOURCES

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This patent application claims priority to, and incorporates by reference in its entirety, U.S. Provisional Patent Application No. 60/473,931, entitled "Method, System and Software for State Signing of Internet Resources" and filed May 28, 2003. This patent application incorporates by reference in its entirety each of the following co-pending U.S. patent applications: 1) "Method and System for Identifying Bidirectional Packet Flow" filed on May 28, 2004; 2) "Policy Based Network Address Translation" filed on May 28, 2004; and 3) "Multilayer Access Control Security System" filed on May 28, 2004.

BACKGROUND

[0002] Internet resources, and in particular web applications, are often developed with fields which can be altered or abused by those accessing a resource. However, the resource developer may assume that the resource is unalterable or secret (unpredictable or incapable of being guessed). These fields can include, for example, cookies, hidden inputs on forms and Uniform Resource Locators ("URLs").

[0003] Many Internet applications, such as web sites and other corporate resources, can be infiltrated using only a text editor. Exemplary Internet attacks that exploit misplaced trust in client data include cookie tampering, parameter tampering, hidden field manipulation, stealth commanding, forceful browsing, Structured Query Language (SQL) injection, overflow attacks, and others application level attacks.

[0004] Current solutions to prevent field tampering exist in the form of software that monitors all traffic going to and from an Internet application. Based on a comprehensive rule set that identifies normal application use, existing products infer session information and analyze all Hypertext Transfer Protocol ("HTTP") packets by putting all packets through a rule engine that represents the web application state machine. A rule set maintained by the rule engine mirrors the complexity of the web application and is derived using either explicit configuration or a "learning mode" which attempts to generate the state machine by observing presumably "correct" behavior.

[0005] Existing solutions have significant drawbacks, however, in terms of configuration complexity, scalability, performance and accuracy. Every web site or web application has a different set of rules that must be learned or configured into the system. Changes to the web site may cause the rule set to be rendered invalid and can require reconfiguration or relearning of a new rule set. The complexity of the rules also requires significant CPU capacity and memory to process each web transaction.

[0006] For the foregoing reasons, a need exists for a method, system and software for state signing to determine whether fields in a web page or other Internet resource have been altered or abused.

SUMMARY

[0007] Before the present methods and systems are described, it is to be understood that this invention is not limited to the particular methodologies and systems described, as these may vary. It is also to be understood that the terminology used in the description is for the purpose of describing the particular versions or embodiments only, and is not intended to limit the scope of the present invention which will be limited only by the appended claims.

[0008] It must also be noted that as used herein and in the appended claims, the singular forms “a,” “an,” and “the” include plural references unless the context clearly dictates otherwise. Thus, for example, reference to a “server” is a reference to one or more servers and equivalents thereof known to those skilled in the art, and so forth. Unless defined otherwise, all technical and scientific terms used herein have the same meanings as commonly understood by one of ordinary skill in the art. Although any methods, materials, and devices similar or equivalent to those described herein can be used in the practice or testing of embodiments of the present invention, the preferred methods, materials, and devices are now described. All publications mentioned herein are incorporated by reference. Nothing herein is to be construed as an admission that the invention is not entitled to antedate such disclosure by virtue of prior invention.

[0009] In an embodiment, a state signing procedure is used to encode the “state” of the web application into the page delivered to the client. In an embodiment, no server side configuration is required. In an alternate embodiment, the only configuration needed on the device (hardware, software, or combination thereof) implementing the state signing function is to enable or disable the state signing for a particular web site. Because a server requires only minimal configuration, web sites may incorporate the state signing system without otherwise modifying the web site or installing software containing complex rules. In an embodiment, the state signing procedure is implemented when the web server transmits the page (either directly or via a web proxy). Changes in the web application do not affect the state signing procedure because the web page is not signed until it is transmitted.

[0010] In an embodiment, state signing is performed by incorporating security metadata in transmissions from a network application server to a client. The security metadata may

contain descriptions describing intended network application semantics. The security metadata may indicate how a URL, cookie, data integrity item (used in form field verification) or other network application element is to be used. At some point subsequent to the receipt of the requested network application element, the client transmits a request back to the server. The server may analyze the transmission from the client to determine if it conforms to the intended network application semantics and is safe to use.

[0011] Intended network application semantics are descriptions of ways in which information can be returned to the server from the client in a form that does not put the server at risk. Examples of intended network application semantics include: returning cookies to servers without modification by the client; clients following links presented to them by the server; clients not modifying the path portions of Uniform Resource Identifiers (“URIs”) before requesting them; clients not modifying queries embedded in the application data; and clients only supplying form data which conforms to the application’s expectations or constraints. Constraints that may be used include the length of the data, type of the data, conformance to types of variables or expected values.

[0012] In an embodiment, a transmission of the application data from an application server is intercepted, proxied, processed or filtered to identify one or more URLs embedded within the transmissions. A predicted form of a client request containing the one or more URLs is generated, indicating how the URL is expected to be requested in the future, and security metadata containing constraints on the manner in which the client requests the one or more URLs is generated. The application data is combined with the security metadata and transmitted to the client. When the client returns the URI and the security metadata via a request, the URI is

validated against the security metadata to ensure that the URI fits within the constraints by which the client may request the URLs.

[0013] In an embodiment, a cryptographic signature is applied over the predicted form of the client request containing the one or more URLs previously transmitted by the application server. The client may return the transmission including the security metadata and the cryptographic signature. The cryptographic signature may be verified to ensure that neither the metadata nor the URI have been altered.

[0014] In an embodiment, a system is generated in which a subsystem receives resource requests from a client to an application server, and a state signing subsystem signs responses to the resource requests delivered to a client from the application server to indicate authenticity. Each of the system and the subsystem may include one or more of hardware and software.

[0015] Enabling/disabling state signing for various data items may be controlled through policies tied to URL prefixes. In an embodiment, individual signatures are accompanied by a control word, embedded into the client side state, and covered by the signature. In an embodiment, the control word includes one or more flags and values which control how the signature is verified when the data item is returned.

BRIEF DESCRIPTION OF THE DRAWINGS

[0016] The accompanying drawings, which are incorporated in and form a part of the specification, illustrate embodiments of the present invention and, together with the description serve to explain the principles of the invention.

[0017] FIGs. 1A and 1B illustrate exemplary direct and indirect client-server requests for application data, respectively, according to embodiments;

[0018] FIG. 2 illustrates an exemplary client-server model using a state signer for application data signing and verification according to an embodiment;

[0019] FIGs. 3A and 3B illustrate exemplary signed URLs and cookies, respectively, according to embodiments;

[0020] FIG. 4 illustrates an exemplary table of signature metadata according to an embodiment;

[0021] FIG. 5 illustrates an example of cookie signing for a set cookie operation according to an embodiment;

[0022] FIG. 6 illustrates an example of cookie signing according to an embodiment.

DETAILED DESCRIPTION

[0023] In describing the embodiments of the invention illustrated in the drawings, specific terminology will be used for the sake of clarity. However, the invention is not intended to be limited to the specific terms so selected, and it is to be understood that each specific term includes all technical equivalents which operate in a similar manner to accomplish a similar purpose.

[0024] Computer program instructions for implementing a software embodiment of the present invention may be stored in a computer program memory or on a computer readable carrier such as a disk, memory stick, portable memory device, communications signal or carrier wave. The steps to complete the methods described herein may be carried out in any computer programming language including object oriented programs including but not limited to Java, C++, Visual Basic, Perl, PHP, and procedural programming languages such as C and Fortran. Other types of programming languages may be utilized, and the invention is not limited to any particular programming language, procedural, object oriented or otherwise.

[0025] Computing platforms for implementing the system include but are not limited to microprocessors and associated peripherals such as external Random Access Memory (RAM), Read Only Memory (ROM) and storage devices including but not limited to magnetic drives, optical and magneto-optical drives, removable storage devices including CD-ROMs, removable magnetic cards, or other types of fixed or removable storage devices. Network interfaces are also included in the computing platform in the form of Ethernet cards, adaptors or other components that allow the transmission and reception of data to and from other computers. Multiprocessor systems and network processors, used individually or coupled together, may be utilized. A computing platform may be constructed from elementary components including a microprocessor and associated peripherals, or may be a pre-assembled computer or computing workstation. The system may also be implemented on custom hardware including Application Specific Integrated Circuits (ASICs) or Field Programmable Gate Arrays (FPGAs). In addition, the system can be embodied in an application server as a web server module, servlets, CGI programs or scripts, server side includes, or directly into the web server source code, application source code either dynamically or statically, or into the operating system itself.

[0026] FIG. 1A illustrates an exemplary configuration for direct client-server requests according to an embodiment. As shown, a client **100** may issue a request **102** to a server **110**. The server **110** may reply to the request **102** with application data **104** and a return request **105**, where the return request **105** is triggered by the application data **104**. The request **102** may be in the form of a URL containing a HTTP address for a resource such as a web page, a File Transfer Protocol (FTP) command or other instruction requesting data, code or other digital information from the server **110**. The server **110** may subsequently return application data **104** to the client **100**. The application data **104** may contain the requested data, code or other digital information.

The application data 104 may alternately include error messages indicating that the requested information is not available.

[0027] Referring to FIG. 1B, a firewall/proxy 120 may be used between the client 100 and the server 110. When the firewall/proxy 120 is used as a firewall, packets that comprise the request 102 may be examined to determine if they contain requests that could be harmful to the server 110. If it is determined that a request 120 is potentially harmful to the server 110, based on a complex set of rules for inspection of the packets, the request 102 may not be forwarded to the server 110.

[0028] When the firewall/proxy 120 is used as a proxy, it may intercept all requests 102 made by the client 100 and determine if those requests are safe for the server 110. If the requests 102 are deemed to be safe, the proxy 120 forwards the request 103 to the server 110 on behalf of the client 100. In both cases, if the request 103 is approved, the application data 104 is returned from the server 110 to the client 100. Subsequent return requests 105 may be subject to the same inspection process to determine if they are safe for the server 110. In all cases, acceptable requests are forwarded to the server 110 without modification. In an embodiment, the firewall/proxy 120 also subjects application data 104 to tests that determine if the application data 104 carries signs of a successful attack. The application data 104 may be dropped if the tests indicate that an attack has been carried out.

[0029] Referring to FIG. 2, the present method, system and software may use a state signer 200 that is placed between the client 100 and the server 110 to receive a request 102 and forward that request 102 to the server 110 as a forwarded request 103. The forwarded request 103 may be a proxy request, with either the state signer 200 or another device acting as the proxy. Upon receiving the forwarded request 103, the server 110 may fulfill the request by

sending application data **104** to the state signer **200**. The state signer **200** may perform a signing operation that results in signed application data **202** being sent to the client **100**. The client **100** may make a signed return request **204** based on the request **102**. The state signer **200** may examine the signed return request **204** to determine if the data received by the client **100** has been tampered with. If the data has not been tampered with, a verified return request **206** may be made to the server **110**. If the state signer **200** does not verify the signed return request **204**, the state signer **200** may, based on its configuration, ignore the request completely, resulting in a dropped request **215**, or reply to client **100** with a denied request **217** bearing an error code. In an embodiment, all requests subsequent to the receipt of signed application data **202** are signed return requests, such as signed return request **204**. In an alternate embodiment, the state signer **200** is configured to verify only certain requests and to sign only certain application data **104**. This discriminatory verification and signing may be accomplished through the use of policies installed into the state signer **200** that select which pieces of application data **104** are signed in the signed application data **202**. These policies may be triggered by or based on URI prefixes associated with each application element, user IDs or other policy predicates. The policies may also be used to determine which requests are in the signed return request **204** as opposed to the unsigned return request form of return request **105**.

[0030] FIG. 3A illustrates a signed URL containing a URL data item **300**, a Control Word (CW) **330**, an Extended Control Word 1 (ECW1) **320**, an Extended Control Word 2 (ECW2) **310**, and a signature **340**. The URL data item **300** may contain a request for resources. The Control Word **330**, the Extended Control Word 1 **320**, and the Extended Control Word 2 **310** may contain metadata indicating one or more aspects of how the URL data item **300** is used. In an embodiment, the URL data item **300** is signed separately from the metadata. In an alternate

embodiment, a query field may indicate a portion of the URI corresponding to user-supplied data. The signature **340** may cover this portion of the request if the query was present in the application data **204** when signing took place. Conversely, if the query represents user-supplied data, the signature **340** does not cover the text of the query. In this instance, ECW1 **320**, for example, may contain a Form Verification Block (FVB) that imposes constraints. The FVB may be covered by the signature **340**, may constrain the data that appears in the query and may apply constraints on the data that appears in the body of the request. CW **330** may include information allowing the state signer **200** to determine which portions of the request are covered by the signature **340** (e.g. query signed v. query not signed) and indicating the presence or absence of FVB information. Similarly, the request **300** may also include a fragment identifier that refers to a subsection of the requested resource. Since the fragment is never returned in a subsequent client request, the signature **340** does not cover the text of the fragment identifier.

[0031] Referring to FIG. 3B, a signed cookie is illustrated in which a cookie data item **350** is combined with metadata, comprising CW **330** and ECW1 **320**, and the signature **340** to produce a cookie item containing additional information such that the state signer **200** can verify its authenticity and safety.

[0032] FIGs. 5 and 6 illustrate cookie-signing examples for a set cookie operation and a return cookie operation, respectively. The upper portion of FIG. 5 illustrates the set cookie header before signing. The lower portion of FIG. 5 represents the set cookie header after signing. In an embodiment, the data items of "PART NUMBER," "BASS_CLARINET," "ITEM_PRICE" and "\$237.14" are signed. In an alternate embodiment, the cookie attributes (e.g. path and secure) may be signed. Referring to FIG. 6, the data items of "CUSTOMER,"

“WILE E COYOTE,” “PART_NUMBER,” “ROCKET LAUNCHER 001,” “SHIPPING” AND “FEDEX” in the upper portion of FIG. 6 are signed.

[0033] Signatures are generated using keys and the text (data fields, metadata fields, or combinations thereof) to be signed. In an embodiment, the signature of a data item is generated using three inputs: UID, text, and key:

- *uid*: the numeric user ID of the requesting user, expressed as an ASCII-encoded hex value, true integer or other numerical representation.
- *text*: holds all text of the object to be protected, and in this embodiment includes signature metadata. For cookies, the text is the single Name/Value (NV) pair appearing after a “Set-Cookie:” or “Set-Cookie2:” literal, excluding reducible white space. In this embodiment, and as illustrated in FIGs. 5 and 6, each NV pair is signed separately and the header arguments are not signed. For URLs, the text may include the fully resolved URL including the protocol, host and full path.
- *key*: a system-wide randomly generated character string, 20 bytes in length, zero-padded out to 64 bytes.

[0034] In an embodiment, a system-wide state signing key, the User ID (UID) associated with the request and the text of the data item (cookie or URL) modified with metadata are concatenated and used as input to a cryptographic signing algorithm. The resulting value from this signing operation may be appended to another permutation of the system-wide state signing key and used as input to the signing algorithm again. In this embodiment, the signature incorporates the identity of the user for whom it was generated in order to prevent the theft of signed elements from caches and cookie files as well as snooping (including proxies such as local proxies, reverse proxies, distributed proxies, corporate proxies, or other proxies).

[0035] In an embodiment, signing a data item means performing the following transformation over its text:

$$\text{sign}(\text{text}, \text{uid}, \text{key}) \Rightarrow$$

$$\text{SHA1_t}(\text{key} \oplus \text{opad} + \text{SHA1_b}(\text{key} \oplus \text{ipad} + \text{uid} + \text{text}))$$

where the following definitions hold:

- The plus operator ('+') is meant to indicate string concatenation;
- The O-plus operator (' $\oplus$ ') is meant to indicate exclusive bitwise OR (XOR) operation;
- **SHA1_t**(*text*) returns the SHA1 digest of its argument, *text*, as a hex string (40 characters);
- **SHA1_b**(*text*) returns the SHA1 digest of its argument, *text*, as a 20 byte number;
- *ipad* = the byte 0x36 repeated 64 times;
- *opad* = the byte 0x5C repeated 64 times; and
- *key* = the encryption key used to generate the secure signature.

This function returns the signature to be appended to the state data.

[0036] In addition to the signature itself, a control word may be incorporated into the item's text before signing it. The control word carries information used by the state signer **200** when later verifying the signature. In an embodiment, the control word is expressed as an ASCII-encoded 3-byte hex value (6 characters).

[0037] FIG. 4 illustrates a table for Control Word **330** content used in one embodiment. CW **330** includes metadata that may indicate the intent of the request, as well as information regarding the signature, how that signature was obtained and a marker facilitating rapid discrimination of signed from unsigned requests.

[0038] In an embodiment, CW 330 has the following grammar:

CTRLWD	=	"F81" KEY QUERY SECURE
KEY	=	"0" for key 0 or "1" for key 1
QUERY	=	"8" for ignore query or "C" for sign query
SECURE	=	"0" for http backend 0 or "8" for https backend

[0039] In the abovementioned embodiment, eight legal hexadecimal values of CW 330 may be used: 0xF81080, 0xF810C0, 0xF81180, 0xF811C0, 0xF81088, 0xF810C8, 0xF81188 and 0xF811C8. In an embodiment, CW 330 appears in a predictable location within any signed text, thus the presence of the text "F81" may be used to distinguish signed content from unsigned content. In this embodiment, extra "marker bits" are included in the control word.

[0040] ECW1 320 and ECW2 310 may be used for FVBs and Signature Expiry Blocks (SEBs), respectively. SEBs may contain an ASCII-encoded date (e.g. ASN.1) beyond which the signature is considered invalid. Each may be used for other purposes as well and may act as extensions to CW 330.

[0041] It is preferable, for security reasons, to update or regenerate the key used for encryption periodically to protect against the possibility of undetected key compromise. In an embodiment, a system-wide key is used rather than per connection, per-application, or per-user keys. In this embodiment, two keys are active in the system at any given time to allow for periodic rotation of the keys. A simple index may be used to track which key is active. In alternate embodiments, a larger number of keys can be used, or the keys can be varied for each connection, application or user. As will be understood by one skilled in the art, a variety of security techniques may be applied for key generation, and public key and/or private key systems may be used.

[0042] Changing or rotating keys may result in a broken client-side state resulting in unusable bookmarks, cookies and/or cached web pages. In an embodiment, the key regeneration interval is set to one month to allow for a maximum two-week window of usability for the client-side state. In addition, the system administrator may cause the state signer **200** to return a “301 Permanent Redirect” message for requests using a key that is valid but not current. This redirect may include a URI signed using the current key. This process may permit bookmarks and cache entries to be updated in many types of Internet browsers including INTERNET EXPLORER® browsers produced by Microsoft Corporation. In this embodiment, bookmarks remain current as long as they are requested at least once every key refresh period. In an alternate embodiment, link verifier software is used to verify the link and refresh broken links with data signed by the current key.

[0043] In an embodiment, keys are 20 bytes long and are cryptographically random. In this embodiment, the keys are generated by a random number generator, such as the random number generator produced by Cavium Networks. Other embodiments with longer or shorter keys are possible, and as previously mentioned, a number of key generation techniques may be utilized.

[0044] In an embodiment, cookie signing can be enabled or disabled. In this embodiment, cookie signing may be controlled on the basis of the URI prefix of the requested URI, as well as the URI prefix of the cookie’s PATH attribute. In this embodiment, if cookie signing is disabled on any URI prefix, the setting is not re-enabled on any URL descendant of that prefix.

[0045] A cookie may not always bear a PATH attribute. For a cookie without a PATH attribute, the standard protocol may be to default to the request-host for the domain and the path

of the request URL that generated the set-cookie response, up to, but not including, the right-most “/.” Preferably, all cookie signing operations occur only on the re-written cookie attributes. In this embodiment, cookie signing takes place after the rewrite on the request headers, but before the rewrite on the response headers. When an incoming Cookie:request header or Set-Cookie:response header is received, a prefix table lookup may be performed over the text of its PATH attribute. If cookie signing is enabled, for that PATH, a valid signature must appear on that cookie. If not, the cookie may be passed through unmodified. A management interface may be provided to ensure that if cookie signing is disabled on any particular prefix, other policies bearing that prefix “inherit” that state.

[0046] With respect to cookie requests, the process may be initiated by consulting a policy database to determine the state of the cookie signing feature (enabled/disabled) for both signature checking and the subsequent signature generation. In an embodiment, the authenticated user ID of the requestor is also examined.

[0047] For cookie bearing requests that need to be verified, a signature may be generated over the user ID and the name value text (plus control word and metadata). The generated signature may be compared to the signature embedded in the returned cookie to determine if the generated signature is indeed verified. If the generated signature is verified, the cookie passes verification. If it is not verified, the cookie, along with its associated attributes, is deleted from the output stream.

[0048] For cookie bearing replies, the state of the cookie signing feature associated with the original URI request may be recalled.

[0049] In a next step for cookie replies, a scan may be performed at the application layer to detect cookie headers and to return the appropriate fields including the cookie name, value, domain, and path. This step may also be performed on cookie requests.

[0050] For identified cookies, a control word may be added. The cookie may then be signed and appropriately inserted into the output stream.

[0051] For URL signing, relative URLs may first be resolved to permit signing. Exemplary URLs that need to be resolved may include network-path references, absolute-path references and relative-path references. Although fully resolved URLs are used in signature generation, the output stream may not need to reference the fully resolved URL.

[0052] In both cookie and URI signing, the state signer 200 may remove the signature, CW 330 and associated metadata from the requested URI and/or cookie before forwarding it to the application server 110.

[0053] Although the invention has been described with reference to the preferred embodiments, it will be apparent to one skilled in the art that variations and modifications are contemplated within the spirit and scope of the invention. The drawings and description of the preferred embodiments are made by way of example rather than to limit the scope of the invention, and it is intended to cover within the spirit and scope of the invention all such changes and modifications.

CLAIMS

What is claimed is:

1. A method for monitoring transmissions between a network application server and a client, the method comprising:
 - incorporating security metadata in a server transmission from a network application server to a client, wherein the security metadata includes a description of the intended network application semantics;
 - receiving a client transmission from the client, wherein the client transmission includes returned metadata; and
 - determining, using the returned metadata, whether the client transmission conforms to the intended network application semantics.
2. The method of claim 1, wherein the security metadata is transmitted with application data.
3. The method of claim 1, wherein the security metadata is transmitted separate from application data.
4. The method of claim 1, wherein the security metadata further comprises authorization information for client transmissions, wherein the authorization information indicates which client transmissions are authorized.

5. A method applied between a network application server and a client to ensure data security:

intercepting a transmission of application data from a network application server to a client;

identifying one or more Uniform Resource Locators (URLs) embedded in the transmission;

generating security metadata containing constraints on the manner in which the client requests the one or more URLs; and

transmitting the transmission of application data and the security metadata to the client.

6. The method of claim 5, further comprising:

receiving a client transmission from the client, wherein the client transmission includes a URL request and the security metadata; and

validating the URL request against the security metadata to ensure that the URL request fits within the constraints on the manner in which the client requests the one or more URLs.

7. The method of claim 5, further comprising generating a cryptographic signature over the predicted form of the client request containing the one or more URLs previously transmitted by the application server, and wherein transmitting further comprises transmitting the cryptographic signature to the client.

8. The method of claim 7, further comprising:

receiving a transmission from the client, wherein the transmission from the client includes a URL request and the security metadata;

validating the URL request against the security metadata to ensure that the URL request fits within the constraints on the manner in which the client requests the one or more URLs; and

verifying the cryptographic signature.

9. An apparatus for monitoring transmissions between a network application server and a client, the apparatus comprising:

a transmitter for incorporating security metadata in one or more server transmissions from a network application server to a client, wherein the security metadata includes a description of the intended network application semantics;

a receiver for receiving one or more client transmissions from the client, wherein the one or more client transmissions include returned metadata; and

an analyzer for determining, using the returned metadata, whether the one or more client transmissions conform to the intended network application semantics.

10. The apparatus of claim 9, wherein the security metadata is transmitted with the application data.

11. The apparatus of claim 9, wherein the security metadata is transmitted separate from the application data.

12. The apparatus of claim 9, wherein the security metadata further comprises authorization information for client transmissions, wherein the authorization information indicates which client transmissions are authorized.

13. An apparatus for insuring data security, comprising:

a receiver for intercepting a transmission of application data from a network application server to a client;

a parser for identifying one or more URLs embedded in the transmission;

a metadata generator for generating security metadata containing constraints on the manner in which the client requests the one or more URLs; and

a transmitter for transmitting the transmission of application data and the security metadata to the client.

14. The apparatus of claim 13, further comprising:

a second receiver for receiving a client transmission from the client, wherein the client transmission includes a URL request and the security metadata; and

a validator for validating the URL request against the security metadata to ensure that the URL request fits within the constraints on the manner in which the client requests the one or more URLs.

15. The apparatus of claim 13, further comprising a generator for generating a cryptographic signature over the predicted form of the client request containing the one or more URLs

previously transmitted by the application server, and wherein the transmitter further transmits the cryptographic signature to the client.

16. The method of claim 15, further comprising:

a second receiver for receiving a transmission from the client, wherein the transmission from the client includes a URL request and the security metadata;

a validator for validating the URL request against the security metadata to ensure that the URL request fits within the constraints on the manner in which the client requests the one or more URLs; and

a verifier for verifying the cryptographic signature.

17. A system for performing state signing of network resources, comprising:

a first subsystem for receiving one or more resource requests from a client to an application server; and

a state signing subsystem for signing responses to resource requests delivered to a client from the application server, wherein the resource requests are signed to indicate authenticity.

18. The system of claim 17, wherein the state signing subsystem includes a cryptographic signature for the generation of a cryptographic signature over at least a portion of the response to the resource request.

19. The system of claim 18, wherein the state signing subsystem further includes a verification subsystem for the determination as to whether a resource request has been altered.

20. A computer program embodied on a computer-readable medium for signing the state of application data transmitted over a network, the computer program comprising:

- a source code segment for intercepting a transmission of application data from an application server to a client;

- a source code segment for identifying one or more URLs embedded in the transmission;

- a source code segment for generating security metadata containing constraints on the manner in which the client requests the one or more URLs;

- a source code segment for transmitting the transmission of application data and the security metadata to the client.

21. The computer program of claim 20, further comprising:

- a source code segment for receiving a client transmission from the client, wherein the client transmission includes a URL request and the security metadata; and

- a source code segment for validating the URL request against the security metadata to ensure that the URL request fits within the constraints on the manner in which the client requests the one or more URLs.

22. The computer program of claim 20, further comprising a source code segment for generating a cryptographic signature over the predicted form of the client request containing the

one or more URLs previously transmitted by the application server, and wherein the source code segment for transmitting further transmits the cryptographic signature to the client.

23. The computer program of claim 22, further comprising:

a source code segment for receiving a client transmission from the client, wherein the client transmission includes a URL request and the security metadata;

a source code segment for validating the URL request against the security metadata to ensure that the URL request fits within the constraints on the manner in which the client requests the one or more URLs; and

a source code segment for verifying the cryptographic signature.

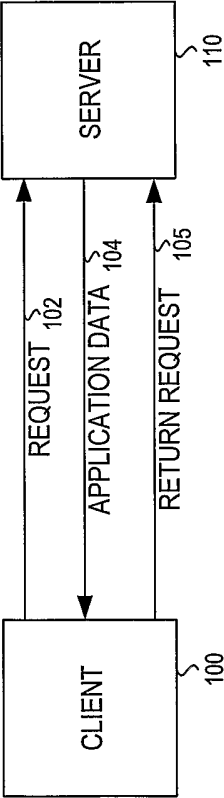


FIG. 1A (PRIOR ART)

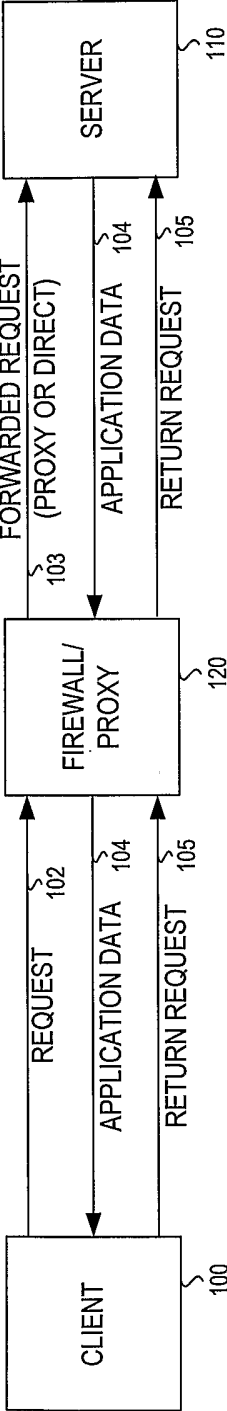


FIG. 1B (PRIOR ART)

2/6

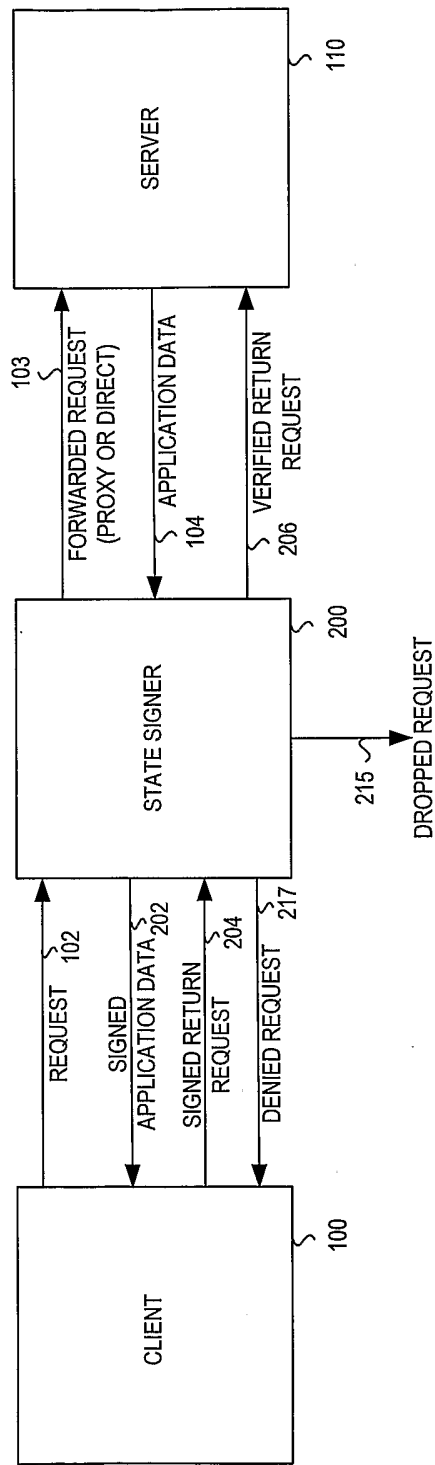


FIG. 2

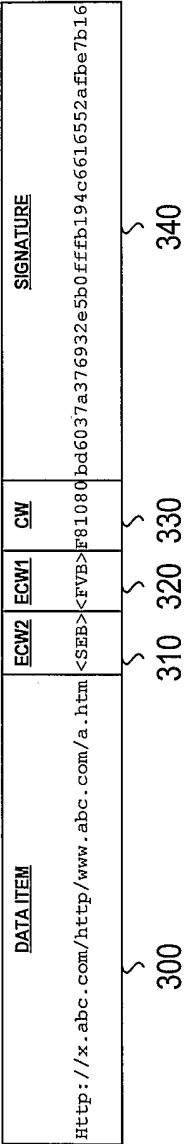


FIG. 3A

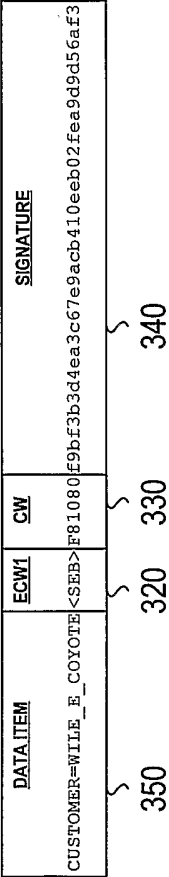


FIG. 3B

4/6

TABLE I CONTROL WORD (CW) FIELDS

FIELD NAME (AND SUB-FIELDS)	LENGTH	VALUES	HEX ENCODING
Marker	8 bits	11111000	F8
Signature Version	4 bits	0001	1
Key selection	4 bits	0000 or 0001	0 or 1
Signature Scope	4 bits	1000 or 1100	8 or C
Sign Over UID (1bit)		1	
Sign Over r Query (1 bit)		0 or 1	
<i>Reserved</i> (2 bits; session?)		00	
Metadata (each 1 bit flags)	4 bits	1000 or 0000	0 or 8
HTTP/HTTPS Backend		0 or 1	
Form Verification Block (FVB)		0	
Signature Expiry Block (SEB)		0	
Extended Control Word (ECW)		0	

FIG . 4

TABLE II COOKIE SIGNING EXAMPLE (SET)

<pre>Set-Cookie:PART_NUMBER=BASS_CLARINET; Domain=extranet.abc.com; Path=/http/www.abc.com , ITEM_PRICE="\$237.14" ; Domain=extranet.abc.com; Path=/http/www.abc.com</pre>
<pre>Set- Cookie:PART_NUMBER=BASS_CLARINETF81080f8373a45df3a5bd727515406e79ca f83879f127c; Domain=extranet.abc.com; Path=/http/www.abc.com , ITEM_PRICE="\$237.14F8108097a7af7c82fff2e3191f1fd63337fcedb121fcfc" ; Domain=extranet.abc.com; Path=/http/www.abc.com</pre>

FIG. 5

TABLE III COOKIE SIGNING EXAMPLE (COOKIE)

<pre>Cookie: CUSTOMER=WILE_E_COYOTE; \$Domain=extranet.abc.com; \$Path=/http/www.abc.com PART_NUMBER="ROCKET LAUNCHER 0001"; \$Domain=extranet.abc.com; \$Path=/http/www.abc.com; SHIPPING = FEDEX; \$Domain=extranet.abc.com; \$Path=/http/www.abc.com</pre>
<pre>Cookie: CUSTOMER=WILE_E_COYOTEf8108088fb4548c4146a7aa4baf2a74d46453fb2b44 b54; \$Domain=extranet.abc.com; \$Path=/http/www.abc.com; PART_NUMBER="ROCKET LAUNCHER 0001f8108063255cbb45dee45e33be2490a5f16575826e4d22"; \$Domain=extranet.abc.com; \$Path=/http/www.abc.com; SHIPPING = FEDEXf810808bd3f97626b2a90b853f12ecbd0e5408e8df4391;~ \$Domain=extranet.abc.com; \$Path=/http/www.abc.com</pre>

FIG. 6