US 20140090054A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2014/0090054 A1

Bolzoni et al. (43) Pub. Date: Mar. 27, 2014

(54) SYSTEM AND METHOD FOR DETECTING ANOMALIES IN ELECTRONIC DOCUMENTS

(75) Inventors: **Damiano Bolzoni**, Enschede (NL); **Emmanuele Zambon**, Enschede (NL)

(73) Assignee: **SECURITYMATTERS B.V.**, Enschede (NL)

(21) Appl. No.: **13/824,211**

(22) PCT Filed: **Jul. 26, 2012**

(86) PCT No.: **PCT/NL12/50537**

§ 371 (c)(1), (2), (4) Date: **Dec. 5, 2013**

Related U.S. Application Data

(60) Provisional application No. 61/511,685, filed on Jul. 26, 2011.

(30) **Foreign Application Priority Data**

Jul. 26, 2011 (NL) 2007180

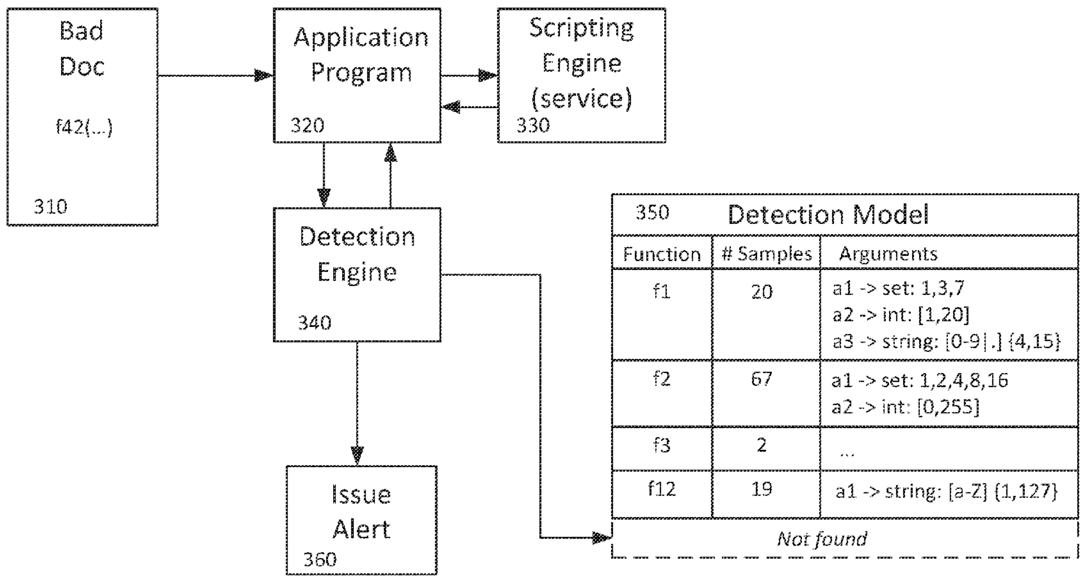
Publication Classification

(51) **Int. Cl.**
G06F 21/50 (2006.01)

(52) **U.S. Cl.**
CPC ***G06F 21/50*** (2013.01)
USPC **726/22**

(57) **ABSTRACT**

A system and method are described herein for detecting an anomaly in an electronic document. In a computer system, a detection engine is attached to an application program which processes the electronic document. Function calls to a service provided through an application program interface (API) are intercepted by the detection engine as the application program processes the electronic document. If an entry for the intercepted function call is not present in the detection model, or an entry is present but the argument value does not match the argument value in the detection model, an alert is raised. The detection model is populated by processing a plurality of known good documents, populating the detection model with entries on intercepted good function calls and their argument values. A threshold may be applied to the detection model, removing from the detection model function calls which were observed less than the threshold amount.



Function	# Samples	Arguments
f1	20	a1 -> set: 1,3,7 a2 -> int: [1,20] a3 -> string: [0-9|.] {4,15}
f2	67	a1 -> set: 1,2,4,8,16 a2 -> int: [0,255]
f3	2	...
f12	19	a1 -> string: [a-Z] {1,127}

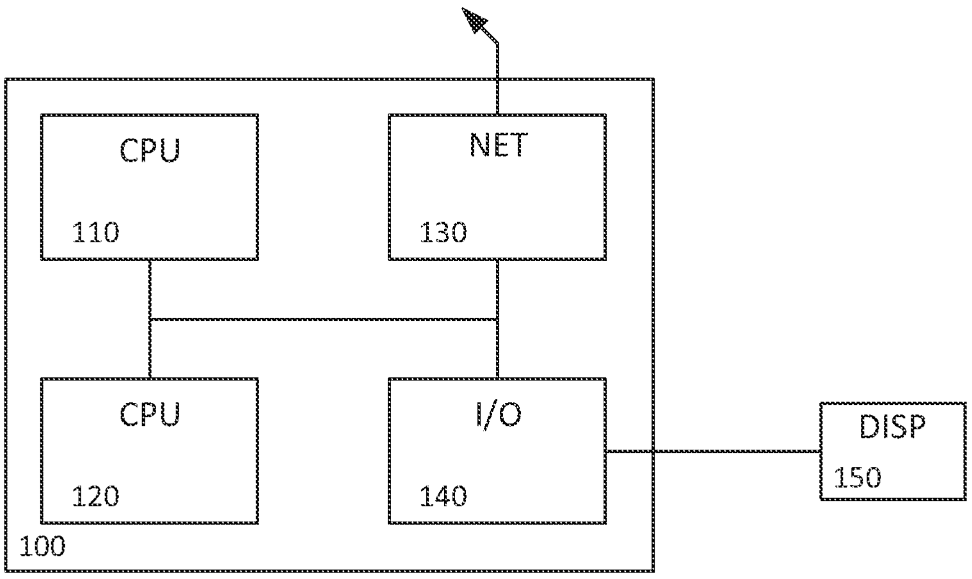


Fig 1

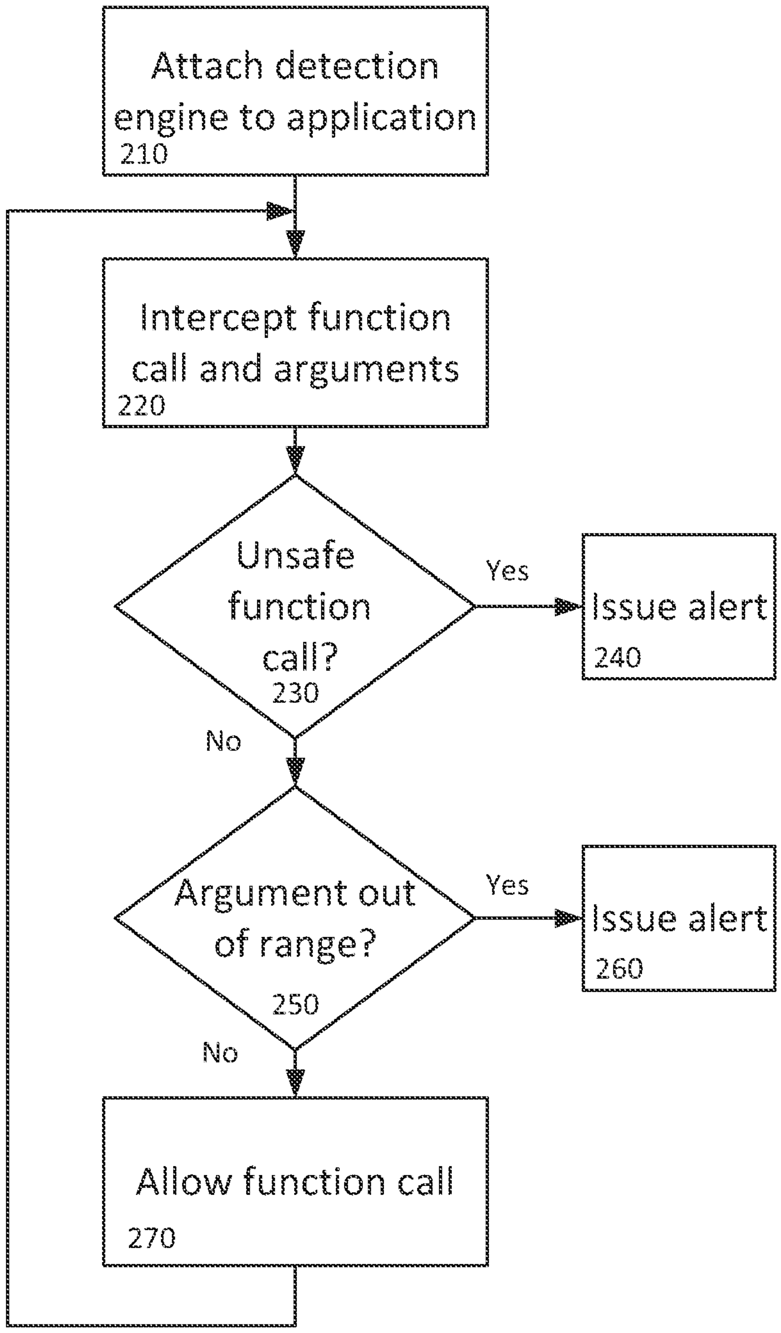
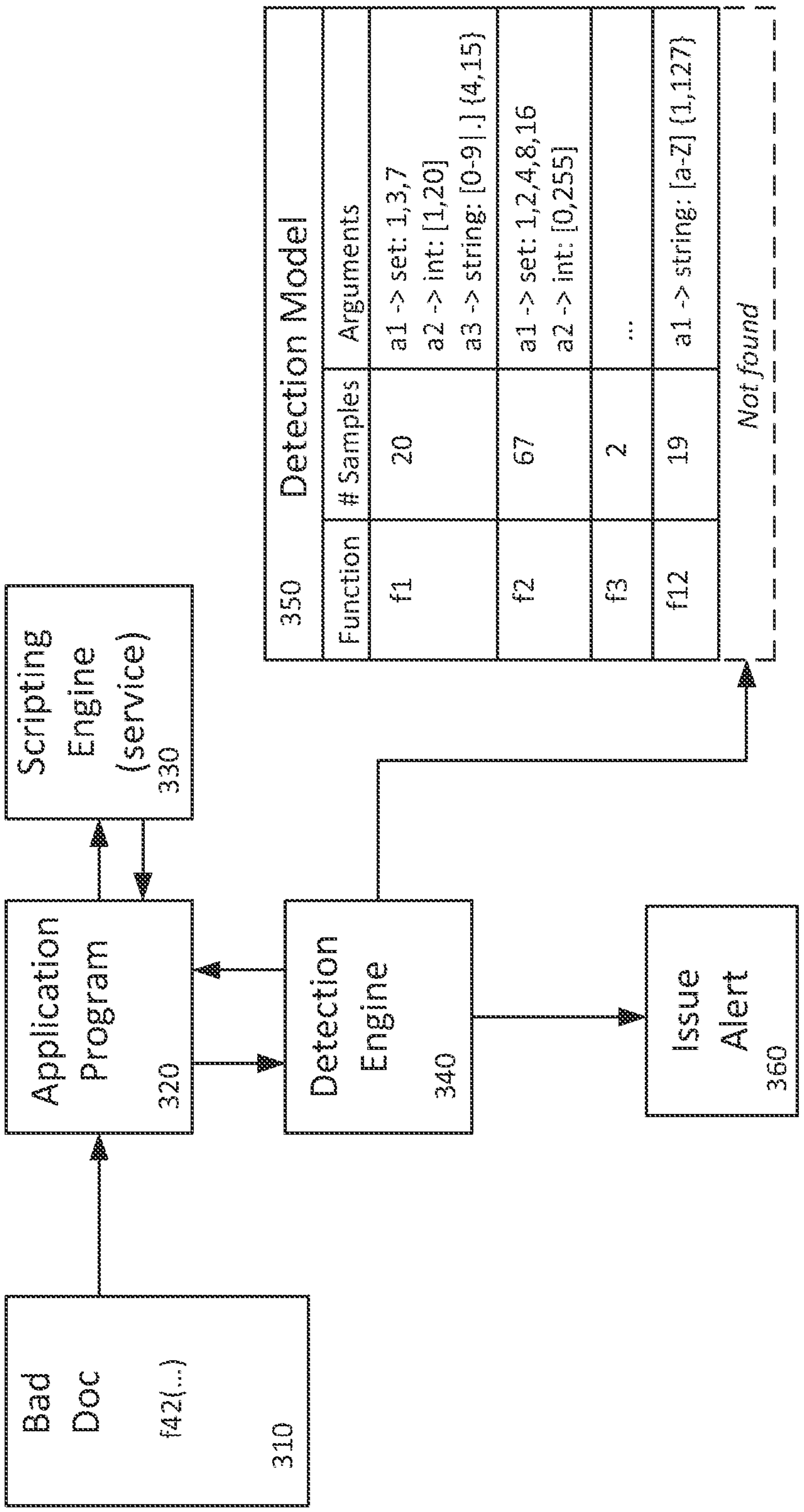
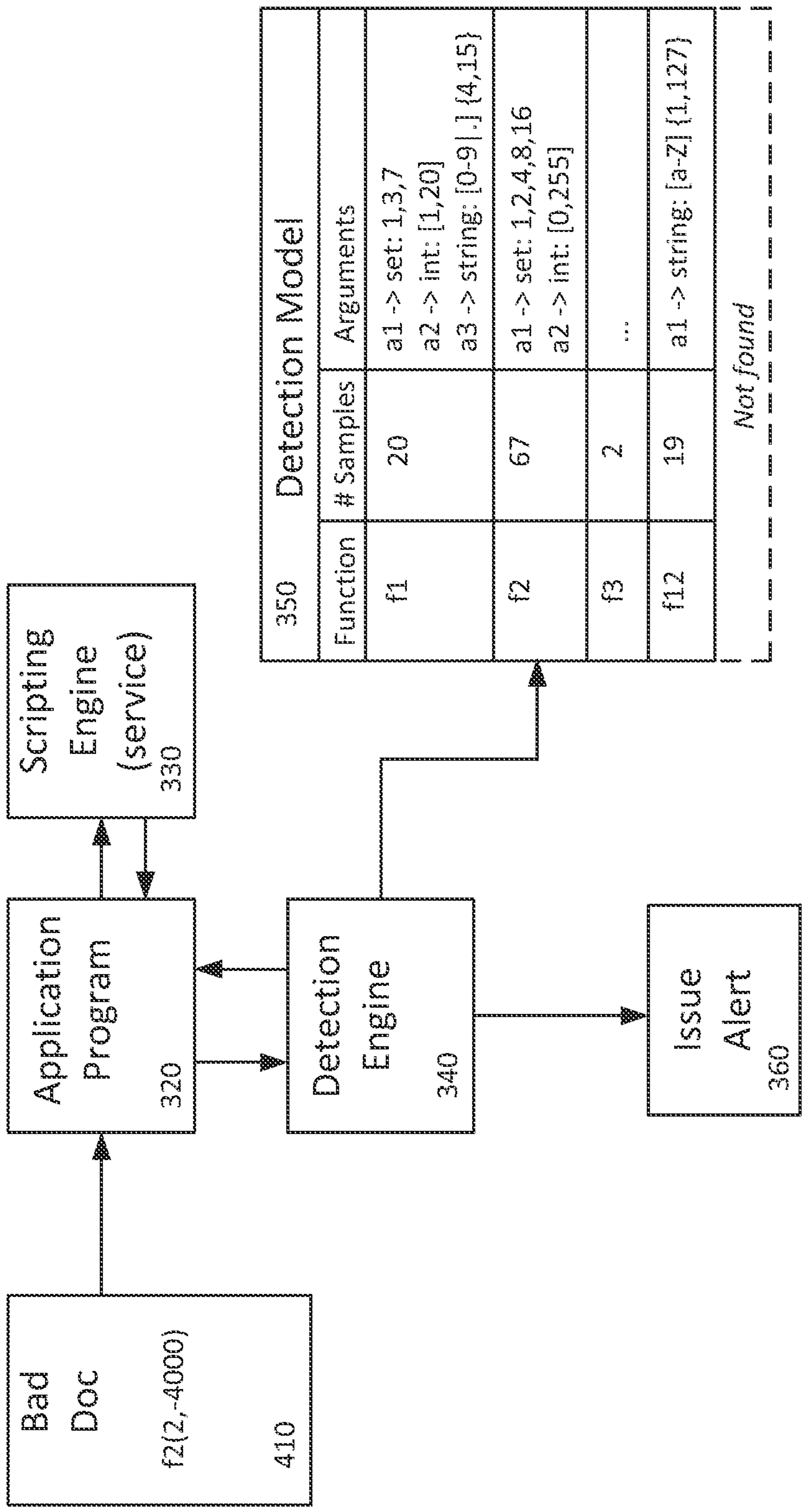

Attach detection engine to application
210
Intercept function call and arguments
220
Unsafe function call?
230
Yes
Issue alert
240
No
Argument out of range?
250
Yes
Issue alert
260
No
Allow function call
270

Fig 2



Function	# Samples	Arguments
f1	20	a1 -> set: 1,3,7 a2 -> int: [1,20] a3 -> string: [0-9|.] {4,15}
f2	67	a1 -> set: 1,2,4,8,16 a2 -> int: [0,255]
f3	2	...
f12	19	a1 -> string: [a-Z] {1,127}

Fig. 3



Function	# Samples	Arguments
f1	20	a1 -> set: 1,3,7 a2 -> int: [1,20] a3 -> string: [0-9|.] {4,15}
f2	67	a1 -> set: 1,2,4,8,16 a2 -> int: [0,255]
f3	2	...
f12	19	a1 -> string: [a-Z] {1,127}
	Not found	

Fig. 4

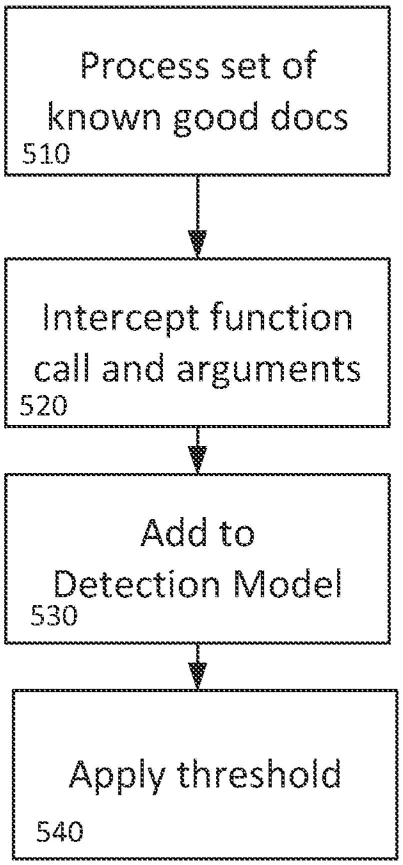

Process set of known good docs
510
Intercept function call and arguments
520
Add to Detection Model
530
Apply threshold
540

Fig 5

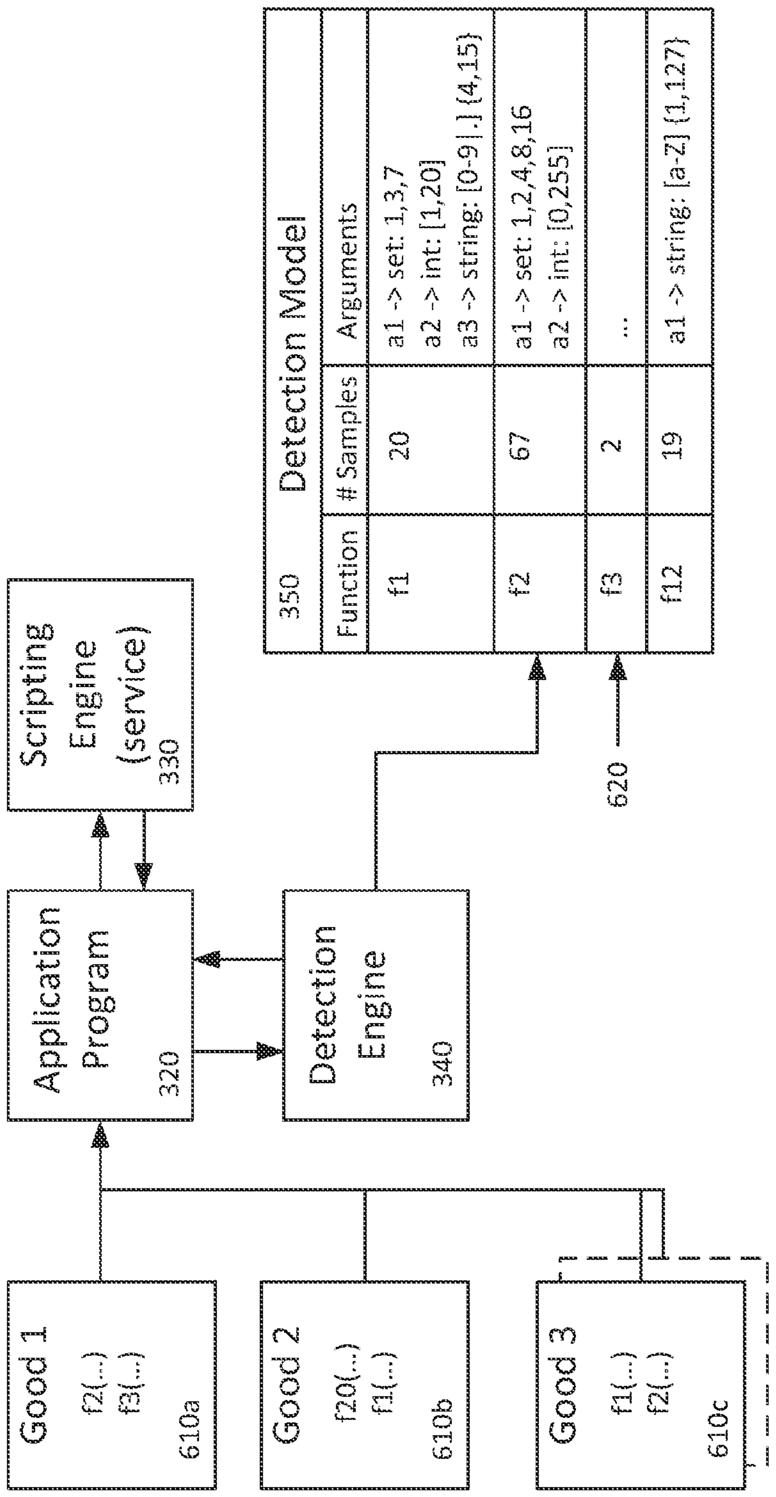

Good 1
f2(...)
f3(...)
610a
Good 2
f20(...)
f1(...)
610b
Good 3
f1(...)
f2(...)
610c
Application Program
320
Scripting Engine (service)
330
Detection Engine
340
350
Detection Model
Function
# Samples
Arguments
f1
20
a1 -> set: 1,3,7
a2 -> int: [1,20]
a3 -> string: [0-9|.]{4,15}
f2
67
a1 -> set: 1,2,4,8,16
a2 -> int: [0,255]
f3
2
...
f12
19
a1 -> string: [a-Z]{1,127}
620

Fig. 6

SYSTEM AND METHOD FOR DETECTING ANOMALIES IN ELECTRONIC DOCUMENTS

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application is related to, and claims priority to, Patent Cooperation Treaty (PCT) International Application Number PCT/NL2012/050537, entitled "METHOD AND SYSTEM FOR CLASSIFYING A PROTOCOL MESSAGE IN A DATA COMMUNICATIONS NETWORK" and filed Jul. 26, 2012, which claims the benefit of U.S. Provisional Patent Application No. 61/511,685, entitled, "METHOD AND SYSTEM FOR CLASSIFYING A PROTOCOL MESSAGE IN A DATA COMMUNICATIONS NETWORK" and filed Jul. 26, 2011, and Netherlands Application No. NL 2007180, entitled "METHOD AND SYSTEM FOR CLASSIFYING A PROTOCOL MESSAGE IN A DATA COMMUNICATIONS NETWORK" and filed Jul. 26, 2011. Each of the aforementioned applications is incorporated herein by reference.

BACKGROUND OF THE INVENTION

[0002] 1. Field of the Invention

[0003] The present invention relates generally to detecting anomalous or malicious content in electronic documents.

[0004] 2. Description of the Prior Art

[0005] Along with the rise of the use of computers in modern life, there has been a rise in the misuse of such computers. One area of this misuse can be referred to as malware, particularly the distribution of electronic documents such as computer files, websites, and the like which have malicious content, usually hidden. Such malware is commonly designed to surreptitiously install programs on a target computer system that allow the target computer system to be exploited remotely, for example capturing keystrokes, accessing files on the target computer system, accessing network connections, and the like.

[0006] Increasingly, malware appears in the guise of seeming innocuous documents, such as web site Hyper Text Markup Language (HTML) documents, documents using the Portable Document Format (PDF) championed by Adobe® Systems and now a standard (ISO 32000-1:2008), documents for Microsoft Office® from Microsoft®, Inc., image documents, and others. These document formats are container formats, as they allow different types of content to be included in one document, for example combining text and graphics with scripting for executing computer programs contained in the electronic document. As examples, PDF and HTML documents support objects of many kinds, including text, graphics, and computer scripting using JavaScript™ or Flash®, all combined into one document.

[0007] When an electronic document such as a PDF document is to be opened on a computer system, the computer operating system activates the application program associated with the electronic document, such as Adobe® Reader® or Adobe Acrobat® from Adobe Systems. The application, for example Adobe Reader, opens the electronic document and interprets the objects the electronic document contains to display the electronic document's contents on a computer screen.

[0008] Unfortunately, a given document may contain not only text and graphics, but also malicious commands which cause a scripting engine such as JavaScript to breach security on the computer system by exploiting software flaws, and surreptitiously install malicious software. Such malicious software can be difficult to detect, and expensive to remedy once present and detected on a computer system.

[0009] Various approaches have been developed to deal with the issues surrounding malicious software, and in preventing malicious software from entering a computer system.

[0010] A widely used approach to malware detection is based on digital signatures of electronic documents. In such a signature-based detection system, the company responsible for the detection system takes a malware-containing electronic document and computes a digital signature for the electronic document. Such digital signature algorithms are well known to the computer arts. The digital signature of the malware containing electronic document is then distributed to the company's customers, where detection software running on target computers computes digital signatures on electronic documents on the computer system, including incoming documents, and compares those signatures to a library of malware signatures, alerting if a match is found, and possibly taking other actions such as quarantining the suspect electronic document.

[0011] Signature-based malware detection systems have a number of serious difficulties. One difficulty is that they only detect malware that has already been identified; they defend against yesterday's known attacks, but not the unknown attacks of tomorrow. An electronic document must have been previously identified as malicious. Then the electronic document must be sent to the company responsible for the detection system. The company verifies the malicious nature of the electronic document, and computes its digital signature. That digital signature is then made available to customers. The updated digital signature must make its way to customer systems, a path fraught with its own difficulties.

[0012] This process of identification, creating a digital signature, and distributing the digital signature to customers may take hours, days, or longer from the time the electronic document is first identified as malicious and submitted to the company. An electronic document may never be submitted as malicious if it is not recognized as malicious; thus a carefully crafted malicious electronic document may continue to be successfully malicious for months or even years.

[0013] Additional difficulties come from the nature of the digital signature process. A digital signature algorithm, related to hashing algorithms in the computer arts, takes an electronic document or computer file and produces a digital signature representing that electronic document or file. As an example, a detection system may create 256-byte digital signatures from electronic documents. Since most electronic documents are larger than this 256-byte signature, mathematically the process is a many-to-one mapping in which at least two different documents having the same 256-byte signature must exist. While digital signature and hashing algorithms are designed to minimize such collisions, mathematically such collisions must exist. In practice in a signature-based malware detection system, when such a collision occurs, the detection system mistakenly identifies an innocuous file as malicious. This is known as a false positive. Instances of false positives are to be minimized as they impede or deny access to valid electronic documents and files.

[0014] An additional difficulty arising from the digital signature process comes from a goal of the digital signature algorithms themselves, that small changes in an electronic document result in large changes in its digital signature. A malware generation or distribution system which introduces a slight variation in each of the malicious electronic documents it delivers thus produces malicious electronic documents each having different digital signatures, thus evading signature-based detection mechanisms.

[0015] Other approaches to dealing with malicious electronic documents and malicious software are anomaly-based, designed in different ways to prevent malware from taking root in a computer system by detecting and preventing malicious behavior as it occurs.

[0016] Some anomaly-based malware detection systems attach themselves to the internals of the computer operating system, monitoring system functions for suspicious behavior. As an example, such a system would alert on an attempt to modify a file marked as belonging to the operating system, or on an attempt to create files in operating system portions of the computer file system. Such anomaly-based systems also have issues with false positive alerts, for example during application program installation or updating, when application program component files must be created or modified.

[0017] What is needed is a better way to detect malicious content in electronic documents.

SUMMARY

[0018] In one embodiment a method of detecting an anomaly in an electronic document comprises: a detection engine intercepting a function call and at least one argument value of the function call, the function call for a service provided through an application program interface (API), the function call generated by an application program processing the electronic document containing the function call, the detection engine determining that the intercepted function call is unsafe by comparing the intercepted function call and the at least one argument value to a detection model, and issuing an alert that an anomaly has been detected in the electronic document.

[0019] In an embodiment, the step of determining that the intercepted function call is unsafe by comparing the intercepted function call and the at least one argument value to the detection model comprises: determining, by the detection engine, that an entry for the intercepted function call is not in the detection model, or determining, by the detection engine, that an entry for the intercepted function call is present in the detection model and that at least one argument value of the intercepted function call does not match a predefined value or range present in the entry for the function call in the detection model.

[0020] In an embodiment, building the detection model comprises: processing, by the application program, a plurality of known good electronic documents, each containing at least one good function call, intercepting, by the detection engine, a good function call of the at least one good function call, and at least one argument value of the good function call, the good function call for the service provided through the API, the good function call generated by the application program processing the known good electronic document containing the good function call, adding, by the detection engine, an entry for the intercepted good function call to the detection model, the entry including the at least one argument value, and repeating the intercepting and adding steps for each good function call in each known good electronic document.

[0021] In an embodiment where the decision model includes at least a number of times the function call is intercepted by the detection engine while building the detection model, a threshold may be applied to the detection model by removing function call entries from the detection model where the number of times the function call was intercepted by the detection engine in processing the plurality of known good electronic documents is less than a threshold, after the plurality of known good electronic documents are processed by the application program.

[0022] In an embodiment, an apparatus for detecting an anomaly in an electronic document comprises: a memory configured to contain a detection model, and a microprocessor coupled to the memory, the microprocessor configured to: intercept a function call and at least one argument value of the function call, the function call for a service provided through an API, the function call generated by an application program processing the electronic document containing the function call, determine that the intercepted function call is unsafe by comparing the intercepted function call and the at least one argument value to the detection model, and issue an alert that an anomaly has been detected in the electronic document.

[0023] In an embodiment, the apparatus for detecting an anomaly in an electronic document is configured to determine that the intercepted function call is unsafe by comparing the intercepted function call and the at least one argument value to the detection model by being further configured to: determine that an entry for the intercepted function call is not in the detection model, or determine that an entry for the intercepted function call is present in the detection model and that the at least one argument value of the intercepted function call does not match a predefined value or range present in the entry for the function call in the detection model.

[0024] In an embodiment, the apparatus for detecting an anomaly in an electronic document is configured to build the detection model contained in the memory by being further configured to: process a plurality of known good electronic documents, each containing at least one good function call, intercept a good function call of the at least one good function call, and at least one argument value of the good function call, the good function call for the service provided through the API, the good function call generated by the application program processing the known good electronic document containing the good function call, add an entry for the intercepted good function call to the detection model contained in the memory, the entry including the at least one argument value, and repeat the intercepting and adding steps for each good function call in each known good electronic document.

[0025] In an embodiment where the decision model includes a number of times the function call is intercepted by the detection engine while building the detection model, the apparatus for detecting an anomaly in an electronic document is further configured to remove function call entries from the detection model contained in the memory where the number of times the function call was intercepted by the detection engine in processing the plurality of known good electronic documents is less than a threshold, after the plurality of known good electronic documents are processed by the application program.

[0026] In an embodiment, a non-transitory computer readable medium having stored thereupon computing instructions for detecting an anomaly in an electronic document comprises: a code segment to intercept a function call and at least one argument value of the function call, the function call for a service provided through an API, the function call generated by an application program processing the electronic document containing the function call, a code segment to determine that the intercepted function call is unsafe by comparing the intercepted function call and the at least one argument value to a detection model, and a code segment to issue an alert that an anomaly has been detected in the electronic document.

[0027] In an embodiment, the non-transitory computer readable medium where the code segment to determine that the intercepted function call is unsafe by comparing the intercepted function call and the at least one argument value to the detection model comprises: a code segment to determine that an entry for the intercepted function call is not in the detection model, and a code segment to determine that an entry for the intercepted function call is present in the detection model and that the at least one argument value of the intercepted function call does not match a predefined value or range present in the entry for the function call in the detection model.

[0028] In an embodiment, the non-transitory computer readable medium having stored thereupon computing instructions for detecting an anomaly in an electronic document further having stored thereupon computing instructions for building the detection model comprising: a code segment to process a plurality of known good electronic documents, each containing at least one good function call, a code segment to intercept a good function call of the at least one good function call, and at least one argument value of the good function call, the good function call for the service provided through the API, the good function call generated by the application program processing the known good electronic document containing the good function call, a code segment to add an entry for the intercepted good function call to the detection model, the entry including the at least one argument value, and a code segment to repeat the intercepting and adding steps for each good function call in each known good electronic document, thereby building the detection model.

[0029] In an embodiment where the code segment to add an entry for the known good function call to the document model includes a code segment to include in the decision model entry a number of times the function call is intercepted by the detection engine while building the detection model, the non-transitory computer readable medium further comprising: a code segment to remove function call entries from the detection model where the number of times the function call was intercepted by the detection engine in processing the plurality of known good electronic documents is less than a threshold, after the plurality of known good electronic documents are processed by the application program.

BRIEF DESCRIPTION OF DRAWINGS

[0030] FIG. **1** is a block diagram of a computer system according to an embodiment.

[0031] FIG. **2** is a flowchart of detecting an anomaly in an electronic document according to an embodiment.

[0032] FIG. **3** is a diagram of detecting an anomaly in an electronic document according to an embodiment.

[0033] FIG. **4** is a diagram of detecting an anomaly in an electronic document according to an embodiment.

[0034] FIG. **5** is a flowchart of building a detection model for use in detecting an anomaly in an electronic document according to an embodiment.

[0035] FIG. **6** is a diagram of building a detection model for use in detecting an anomaly in an electronic document according to an embodiment.

DETAILED DESCRIPTION OF THE INVENTION

[0036] A common vehicle for distributing malware is the use of malicious electronic documents which outwardly appear to be innocuous and of interest to a user, but contain embedded commands to exploit a vulnerability in the software running on the computer system and install malicious software or perform some malicious action. For example, a user may receive a document attached to an electronic mail message, the document having a title such as "QuarterlyBonusInfo.PDF" or "WeekendPartyPics.PDF" which may contain some legitimate content, but also contains function calls generated by an application program such as Adobe Reader for a service provided through an API such as JavaScript, Flash, or a dynamically linked library (DLL) to exploit a vulnerability in the system software, and install malicious software.

[0037] Presented herein are various embodiments of a method and apparatus for detecting such malicious content in electronic documents.

[0038] As described previously herein, signature-based malware detection at best protects against yesterday's attacks, and is easily circumvented.

[0039] Accordingly, a method and apparatus are described in which a detection engine monitors an application program processing an electronic document, for example Adobe Reader processing a PDF electronic document. The detection engine is a separate software component which intercepts function calls from the separate application program to a service provided through an API as the application program processes the electronic document, and determines if those function calls represent anomalies which should result in an alert being issued.

[0040] The application program processes the electronic document. The electronic document may contain function calls to a service provided through the API provided by the application program. These function calls may be present in the electronic document in the form of text, or encoded in binary or other suitable representation. The application program takes the function call present in the electronic document and generates the function call to the service provided through the API to execute the function call. The application program generates the function call from the electronic document for example by translating the text or encoded function call in the document to the form required for the API. This process will be determined by the requirements of the API, and may involve processes known to the computer arts such as tokenizing, table look-ups, compiling, interpretation, or the like to generate a function call to the service provider as required by the API.

[0041] As is known in the art, an API provides a mechanism for an application program to make use of services provided by other computer programs such as scripting engines, dynamic linked libraries (DLLs), dynamic libraries (dylibs), ActiveX control, shared object files (so), and the like. The API mechanism allows a computer program stored as one computer file, e.g. an application program, to make use of services provided by a computer program stored as another computer file, e.g., a scripting engine, DLL, dylib, ActiveX control, or the like. Unix, Linux, and Apple® Macintosh® OSX computer systems use the API mechanism to provide services such as scripting engines (e.g., JavaScript, Flash or Visual Basic), system extensions such as device drivers and kernel extensions, and shared libraries. Microsoft Windows systems use the API mechanism to provide services such as scripting engines (e.g. JavaScript, Flash and Visual Basic), dynamic linked libraries, and system extensions such as device drivers and ActiveX controls.

[0042] As an example, Adobe Reader provides APIs to make use of services provided by the JavaScript scripting engine. By providing services through an API, both the application program making use of the services and the service provider can be maintained and upgraded separately and independently. As an example, the JavaScript scripting engine can be updated to provide additional functionality or to fix programming bugs without having to modify Adobe Reader. Similarly, Adobe Reader can be updated without having to modify JavaScript, as the API provides access to services in a manner independent of the versions of the application programs or services.

[0043] By attaching to APIs, for example the APIs provided by Adobe Reader, the detection engine intercepts function calls from the application program through the API to a service provided through the API, e.g., the JavaScript scripting engine. The function calls are generated by the application program as it interprets an electronic document. The detection engine determines if those function calls represent anomalies which should result in an alert being issued.

[0044] In operation, an application program such as a web browser (e.g., Firefox®, Chrome®, Safari®, Internet Explorer®, or the like) processes text from an electronic document such as a web page (i.e. a HTML document). When the web browser processes text marked as JavaScript, for example a fragment such as "parseFloat(kstr)", the web browser generates a function call to the JavaScript parseFloat( ) function, passing the string argument kstr. The JavaScript scripting engine processes this function call to the parseFloat function and returns a floating point number. An application program such as Adobe Reader goes through the same process, taking text contained in an electronic document and generating, from text marked as JavaScript, function calls to the JavaScript scripting engine. An application program such as Microsoft Word or Microsoft Excel, components of Microsoft Office, go through the same process in taking text marked as Visual Basic, for example in a macro contained in an electronic document such as a spreadsheet or word processing document, and generating function calls to the Visual Basic scripting engine.

[0045] The detection engine uses a detection model which is built by causing the application program to process a set of known good electronic documents. As the set of electronic documents, for example a set of PDF electronic documents, are known to be good, the commands, such as function calls and argument values to the function calls for the services provided through an API, e.g., function calls to JavaScript, contained in these known good electronic documents are also assumed to be good. The detection engine populates the detection model with entries generated by these known good documents, building entries on observed function calls and observed argument values contained in these known good electronic documents.

[0046] In operation, that is, after the detection model has been built, as the application program processes an electronic document, the detection engine intercepts function calls and their arguments generated by the application program as it processes the electronic document prior to those function calls being passed from the application program to the scripting engine.

[0047] The terms "safe" and "unsafe" are used herein to refer to the determinations by the detection engine with respect to a particular function call and the arguments to that function call. These determinations of safe or unsafe indicate a possible attempt to exercise a vulnerability leading to a compromise of the computer system. As such, they are distinct from the use of a function call and its arguments in a computer programming sense. For example, a particular function call may be defined as a legitimate function call in a particular service provided through an API such as JavaScript, Flash, or in a particular DLL, and thus available for use by programmers, but if an entry for that function call does not exist in the detection model, indicating that the function call was not observed in the set of known good electronic documents, the detection engine will consider that function call to be unsafe. When such a condition occurs, the detection engine issues an alert that an anomaly has been detected in the electronic document.

[0048] Similarly, if an entry for the intercepted function call is present in the document model, the argument values of the intercepted function call are tested, and if an argument value does not match or is outside the range for that argument contained in the detection model entry for the intercepted function, the intercepted function call is likewise considered unsafe. When such a condition occurs, the detection engine issues an alert that an anomaly has been detected in the electronic document.

[0049] An alert can include one or more of: a visual display on a computer screen such as an alert dialog box, logging the anomaly on the computer system or through a logging service, or aborting further processing of the electronic document by the application program.

[0050] Turning now to FIG. **1**, a block diagram of a computer system **100** which may be used to practice the invention is shown in simplified form. As understood in the art, computer system **100** comprises a central processing unit (CPU) **110** which is coupled to memory hierarchy **120**, network interface **130**, and Input/Output (I/O) interface **140**. CPU **110** may be a microprocessor such as an x86 class processor from Intel Corporation or Advanced Micro Devices. Other microprocessors such as those offered by MIPS, Advanced Risc Machines (ARM), and others may also be used.

[0051] Memory hierarchy **120**, as understood by the art includes any combination of a permanent memory device for use in initializing the computer system on power-up, fast read-write main memory such as Random Access Memory (RAM) for holding instructions and data for use by microprocessor **110**, and file storage devices including but not limited to flash memory, disc drives including solid state disks, memory cards and the like, for storing electronic documents which include operating system files, programs including applications programs, and data files for use by the computer system.

[0052] Network interface **130** may include wired and wireless interfaces such as those compatible with IEEE 802.3 wired Ethernet standards or IEEE 802.11 WiFi standards, and connects to local and/or wide area networks, not shown. Input/Output interface **140** may include support for keyboards and graphic input devices such as mice and tablets, and output devices such as a display shown as DISP **150**.

[0053] Computer system **100** operates under the control of an operating system, such as Microsoft Windows from Microsoft Corporation, OS/X from Apple Computer, or one of the many open-source Linux operating systems.

[0054] Referring now to the flowchart of FIG. **2** and the diagrams of FIGS. **3** and **4**, operation of the detection engine in detecting an anomaly in an electronic document according to an embodiment will now be described.

[0055] In step **210**, the detection engine is attached to an application. As is known in the art, the application program processes electronic documents of a particular document type, for example, Adobe Reader processes PDF electronic documents, sending function calls and arguments for those function calls to a service provided through an API such as JavaScript. In an embodiment, the detection engine attaches itself to the application to intercept function calls from the application program to a scripting engine.

[0056] In an embodiment, the Adobe Reader application program from Adobe Systems running on a Windows® operating system such as Windows 7 from Microsoft® Corporation, uses an API to access services provided by the JavaScript scripting engine. Other services provided through similar APIs which may be supported in other embodiments include Java® from Oracle Corporation, Adobe Flash® the open-source Python, or Visual Basic from Microsoft Corporation.

[0057] As is understood in the art, the process of attaching a software component such as the detection engine to an application program such as Adobe Reader is dependent on the operating system on which the detection engine and application program run. In the embodiment described, Adobe Reader running on a Windows operating system, Adobe Reader provides application program interfaces (APIs) to a MethodDispatcher and an ArgumentParser. Detection engine **340** attaches computer code in detection engine **340** to the Adobe Reader MethodDispatcher API to intercept function calls from application program **320**, Adobe Reader, to scripting engine **330**, JavaScript. Detection engine **340** attaches computer code in detection engine **340** to the Adobe Reader ArgumentParser API to retrieve argument values for the intercepted function call.

[0058] In step **220** of FIG. **2**, detection engine **340** intercepts a function call and arguments for the function call from application program **320** to scripting engine **330**. In an embodiment, electronic document **310** may contain many different objects. In the case of a PDF electronic document, these objects include text, graphics, and scripting instructions to a scripting engine such as JavaScript, including function calls to JavaScript functions. As application program **320**, in the embodiment, Adobe Reader, interprets the contents of electronic document **310**, application program **320** turns these scripting instructions into function calls to be sent from application program **320** to scripting engine **330**.

[0059] In an embodiment, function calls from application program **320** to scripting engine **330** are intercepted by detection engine **340** using the Adobe Reader MethodDispatcher API. Detection engine **340** retrieves the argument values for the intercepted function call using the Adobe Reader ArgumentParser API.

[0060] In step **230**, the detection engine determines if the intercepted function call is unknown to the detection model. In an embodiment, detection engine **340** determines if an entry for the intercepted function call is present in detection model **350**. If no entry for the intercepted function call is present in detection model **350**, the function call is deemed unsafe.

[0061] This process is shown in more detail in FIG. **3**. As shown, electronic document **310** contains a function call f42(. . .). In processing electronic document **310**, application program **320** generates a function call to scripting engine **330**. This function call is intercepted by detection engine **340**. As shown, detection model **350** contains entries for f1 through f12, but does not contain an entry for f42. Thus the intercepted function call f42 is deemed unsafe by detection engine **340**.

[0062] In step **240**, if no entry for the intercepted function call is present in the detection model, an alert is issued by detection engine **340** indicating that an anomaly has been detected in the electronic document.

[0063] Issuing an alert may include one or more of displaying an alert on a computer display, logging the alert, or aborting processing of the electronic document by the application program. In an embodiment, an alert dialog box may be displayed to a user indicating an anomaly has been detected in the electronic document. The alert may be logged, such as to a log file on the computer system, or through a network-based logging mechanism. Further processing of the electronic document by the application may be aborted. In an embodiment, these alert options may be configurable, for example, by a user or a management service.

[0064] In step **250**, a determination is made regarding whether the argument values of the intercepted function call matches a known value or are out of range. If an entry in the detection model is present for the intercepted function call, which has been determined to be present in Step **230**, the argument values for the intercepted function call are matched against values and/or ranges present for the arguments in the detection model entry for the intercepted function call. If any argument values do not match the values and/or ranges present in the detection model, the function call is deemed unsafe by detection engine **340**.

[0065] This is shown in more detail in FIG. **4**. As shown, electronic document **410** contains a function call f2(2, −4000). In processing electronic document **410**, application program **320** generates a function call to scripting engine **330**. This function call is intercepted by detection engine **340**. As shown, detection model **350** contains an entry for f2. The entry for f2 in detection model **350** shows two arguments, first argument a1 with valid values from the set 1, 2, 4, 8, 16 and second argument a2 with valid values in the integer range 0 to 255. In the intercepted function call, the first argument to function f2 is 2, which matches the set and is valid. The second argument is −4000, which is out of the integer range 0 to 255, and is unsafe. Thus the intercepted function call f2(2, −4000) is deemed unsafe by detection engine **340**.

[0066] In step **260** if an argument for the intercepted function call does not match or is out of range when compared to the entry in detection model **350** for the function call, an alert is issued by detection engine **340** indicating that an anomaly has been detected in the electronic document.

[0067] In step **270**, because the intercepted function call and argument values to the intercepted function call have been determined to be valid by Steps **230** and **250**, the intercepted function call is allowed to proceed to scripting engine **330**.

[0068] In an embodiment, this process is repeated each time a function call from the application program to the scripting engine is generated by the application program processing the electronic document.

[0069] FIG. **5** shows a flowchart for building a detection model for use in detecting an anomaly in an electronic document according to an embodiment.

[0070] In step **510**, the detection model is built by populating the detection model with known good function calls generated by processing a plurality of known good electronic documents by the application program. In an embodiment, referring to FIG. **6**, a set of known good documents, **610***a*, **610***b*, **610***c* and so on are processed by application program **320**. In an embodiment, the detection engine is operated in a detection model building mode for a period of time, such as a period of hours, e.g., twenty four hours, during which known good electronic documents are processed by the application program.

[0071] In step **520** of FIG. **5** the detection engine **340** intercepts a function call and argument values from application program **320** to scripting engine **330** as application program **320** processes a known good document.

[0072] In step **530**, an entry on the intercepted function call and its argument values are added to detection model **350**.

[0073] In an embodiment, added to detection model **350** means that if an entry for the intercepted known good function call is not present in detection model **350**, an entry for the intercepted function is added. This entry includes observed argument values. Similarly, if an entry already exists for this intercepted known good function call, argument values for the intercepted known good function call are combined with the argument values previously added to the detection model entry. As an example, with integer arguments, values are accumulated as sets or ranges. For strings, information such as allowable characters and string lengths are accumulated.

[0074] In an embodiment, a count of the number of times this intercepted function has been observed is also part of the entry in detection model **350**; for the first time this intercepted function is observed, this count is set to 1. When this intercepted function is subsequently observed, the count in detection model **350** for this function is incremented.

[0075] At the completion, for example of the period of time, when all electronic documents **610** in the set of known good electronic documents have been processed by application program **320**, with detection engine **340** populating detection model **350** with entries on intercepted good function calls from known good electronic documents **610** as processed by application program **320**.

[0076] As an example of populating detection model **350**, referring to FIG. **6**, known good electronic document **610***a* contains known good function calls to f2 and f3. Known good electronic document **610***b* has known good function calls f20 and f1. Known good electronic document **610***c* has known good function calls f1 and f2, and so on through the set of known good documents.

[0077] In an embodiment where a count of the number of times a function has been observed is kept in detection model **350**, in step **560**, a threshold is applied to detection model **350**, removing entries for function calls if the number of times the function call was intercepted in processing the plurality of known good electronic documents **610** is below the threshold. The threshold is applied in such an embodiment based on the premise that the threshold insures that a valid sample size of intercepted function calls for the particular function call have been obtained.

[0078] Referring again to FIG. **6**, in an embodiment where a count of the number of times a function has been observed is kept in detection model **350**, assume that the threshold is 10. As shown in detection model **350**, function f1 was intercepted 20 times, function f2 was intercepted 67 times, function f3 was intercepted 2 times, and function f12 was intercepted 19 times. Of these entries, function f3 is below the threshold of 10, and the entry for function f3 is therefore removed from detection model **350**.

[0079] It should be noted that in an embodiment where a count of the number of times a function has been intercepted is kept in detection model **350**, this count data is only used during the detection model building phase, and is not needed for the operation of the detection engine in detecting anomalies in electronic documents. As such, the count data could be removed from detection model **350** after detection model **350** is populated and the threshold of step **560** has been applied.

[0080] While the disclosed method and apparatus has been explained with respect to particular embodiments, such as using Adobe Reader and PDF electronic documents containing scripting in JavaScript, other embodiments will be apparent to those skilled in the art in light of this disclosure, including but not limited to processing of Flash embedded in PDF electronic documents, HTML documents by web browsers such as Internet Explorer, processing of Microsoft Office documents by Microsoft Office, and the like.

[0081] As described previously herein, a scripting engine such as JavaScript is an example of a service provided through an API. Certain aspects of the described method and apparatus may be readily implemented, for example, with application programs using services provided through APIs such as dynamically linked libraries (DLLs) to extend the functionality of the application program. Examples include but are not limited to ActiveX controls on Microsoft Windows operating systems, Java DLLs such as JAR files, and shared object (so) and dynamic library (dylib) files on Unix, Linux and Apple® Macintosh® OSX operating systems. Application programs making use of services provided by APIs include but are not limited to Adobe Acrobat, Adobe Reader, Microsoft Internet Explorer, and Microsoft Office.

[0082] Certain aspects of the described method and apparatus may readily be implemented using configurations other than those described in the embodiments above, or in conjunction with elements other than those described above. For example, the methods may be practiced on a wide range of computing equipment, including but not limited to servers, desktop computers, virtualized systems, embedded systems, and portable devices such as laptops, tablets, smart phones, appliances, and other devices containing embedded computer systems which may use, process, display, or transport electronic documents which may have anomalous or malicious content, operating under operating systems including Windows operating systems from Microsoft Corporation, OSX and iOS operating systems from Apple Inc, Unix, or Linux operating systems among others.

[0083] Further, it should also be appreciated that the described method and apparatus can be implemented in numerous ways, including as a process, an apparatus, or a system. The methods described herein may be implemented by program instructions for instructing a processor to perform such methods, and such instructions recorded on a non-transitory computer readable storage medium such as a hard disk drive, floppy disk, optical disc such as a compact disc (CD) or digital versatile disc (DVD), flash memory, memory cards, etc., or a computer network wherein the program instructions are sent over optical or wired or wireless electronic communication links. It should be noted that the order of the steps of the methods described herein may be altered and still be within the scope of the disclosure.

[0084] It is to be understood that the examples given are for illustrative purposes only and may be extended to other implementations and embodiments with different conventions and techniques. While a number of embodiments are described, there is no intent to limit the disclosure to the embodiment(s) disclosed herein. On the contrary, the intent is to cover all alternatives, modifications, and equivalents apparent to those familiar with the art.

[0085] In the foregoing specification, the invention is described with reference to specific embodiments thereof, but those skilled in the art will recognize that the invention is not limited thereto. Various features and aspects of the above-described invention may be used individually or jointly. Further, the invention can be utilized in any number of environments and applications beyond those described herein without departing from the broader spirit and scope of the specification. The specification and drawings are, accordingly, to be regarded as illustrative rather than restrictive. It will be recognized that the terms "comprising," "including," and "having," as used herein, are specifically intended to be read as open-ended terms of art.

1-34. (canceled)

35. A method of detecting an anomaly in an electronic document comprising:

intercepting, by a detection engine, a function call and at least one argument value of the function call, the function call for a service provided through an application program interface, the function call generated by an application program processing the electronic document containing the function call,

determining, by the detection engine, that the intercepted function call is unsafe by comparing the intercepted function call and the at least one argument value to a detection model, and

issuing an alert, by the detection engine, that an anomaly has been detected in the electronic document.

36. The method of claim **35** where the service provided through the application program interface is a scripting engine.

37. The method of claim **35** where the step of issuing an alert by the detection engine comprises one or more of:

displaying the alert,

logging the alert, or

aborting any further processing of the electronic document by the application program.

38. The method of claim **35** where the step of determining, by the detection engine, that the intercepted function call is unsafe by comparing the intercepted function call and the at least one argument value to the detection model comprises:

determining, by the detection engine, that an entry for the intercepted function call is not in the detection model, or

determining, by the detection engine, that an entry for the intercepted function call is present in the detection model and that the at least one argument value of the intercepted function call does not match a predefined value or range present in the entry for the function call in the detection model.

39. The method of claim **35** further comprising building the detection model by:

processing, by the application program, a plurality of known good electronic documents, each containing at least one good function call,

intercepting, by the detection engine, a good function call of the at least one good function call, and at least one argument value of the good function call, the good function call for the service provided through the application program interface, the good function call generated by the application program processing the known good electronic document containing the good function call,

adding, by the detection engine, an entry for the intercepted good function call to the detection model, the entry including the at least one argument value, and

repeating the intercepting and adding steps for each good function call in each known good electronic document.

40. The method of claim **39** where the step of adding, by the detection engine, an entry for the intercepted good function call to the detection model further comprises: including a number of times the function call is intercepted by the detection engine while building the detection model.

41. The method of claim **40** further comprising:

removing, by the detection engine, function call entries from the detection model where the number of times the function call was intercepted by the detection engine in processing the plurality of known good electronic documents is less than a threshold, after the plurality of known good electronic documents are processed by the application program.

42. The method of claim **35** further comprising:

attaching, by the detection engine, the detection engine to the application program,

before the step of intercepting, by the detection engine, the function call and at least one argument value of the function call

43. An apparatus for detecting an anomaly in an electronic document comprising:

a memory configured to contain a detection model, and

a microprocessor coupled to the memory, the microprocessor configured to:

intercept a function call and at least one argument value of the function call, the function call for a service provided through an application program interface, the function call generated by an application program processing the electronic document containing the function call,

determine that the intercepted function call is unsafe by comparing the intercepted function call and the at least one argument value to the detection model, and

issue an alert that an anomaly has been detected in the electronic document.

44. The apparatus of claim **43** where the microprocessor is further configured to issue an alert by one or more of:

displaying a visible alert on a display coupled to the microprocessor,

logging the alert to a file stored in the memory, or

aborting any further processing of the electronic document by the application program.

45. The apparatus of claim **43** where the microprocessor is configured to determine that the intercepted function call is unsafe by comparing the intercepted function call and the at least one argument value to the detection model by being further configured to:

determine that an entry for the intercepted function call is not in the detection model, or

determine that an entry for the intercepted function call is present in the detection model and that the at least one argument value of the intercepted function call does not match a predefined value or range present in the entry for the function call in the detection model.

46. The apparatus of claim **43** where the microprocessor is configured to build the detection model contained in the memory by being further configured to:

process a plurality of known good electronic documents, each containing at least one good function call,

intercept a good function call of the at least one good function call, and at least one argument value of the good function call, the good function call for the service provided through the application program interface, the good function call generated by the application program processing the known good electronic document containing the good function call,

add an entry for the intercepted good function call to the detection model contained in the memory, the entry including the at least one argument value, and

repeat the intercepting and adding steps for each good function call in each known good electronic document.

47. The apparatus of claim **46** where the microprocessor is configured to add an entry for the intercepted good function call to the detection model contained in the memory bye being further configured to include in the entry at least a number of times the function call is intercepted by the detection while building the detection model.

48. The apparatus of claim **47** where the microprocessor is further configured to:

remove function call entries from the detection model contained in the memory where the number of times the function call was intercepted by the detection engine in processing the plurality of known good electronic documents is less than a threshold, after the plurality of known good electronic documents are processed by the application program.

49. The apparatus of claim **43** where the microprocessor is further configured to attach the detection engine to the application program, before intercepting the function call.

50. A non-transitory computer readable medium having stored thereupon computing instructions for detecting an anomaly in an electronic document comprising:

a code segment to intercept a function call and at least one argument value of the function call, the function call for a service provided through an application program interface, the function call generated by an application program processing the electronic document containing the function call,

a code segment to determine that the intercepted function call is unsafe by comparing the intercepted function call and the at least one argument value to a detection model, and

a code segment to issue an alert that an anomaly has been detected in the electronic document.

51. The non-transitory computer readable medium of claim **50** where the code segment to issue an alert by the detection engine further comprises a code segment to:

display a visible alert,

log the alert, or

abort any further processing of the electronic document by the application program.

52. The non-transitory computer readable medium of claim **50** where the code segment to determine that the intercepted function call is unsafe by comparing the intercepted function call and the at least one argument value to the detection model comprises:

a code segment to determine that an entry for the intercepted function call is not in the detection model, and

a code segment to determine that an entry for the intercepted function call is present in the detection model and that the at least one argument value of the intercepted function call does not match a predefined value or range present in the entry for the function call in the detection model.

53. The non-transitory computer readable medium having stored thereupon computing instructions for detecting an anomaly in an electronic document of claim **52** further having stored thereupon computing instructions for building the detection model comprising:

a code segment to process a plurality of known good electronic documents, each containing at least one good function call,

a code segment to intercept a good function call of the at least one good function call, and at least one argument value of the good function call, the good function call for the service provided through the application program interface, the function call generated by the application program processing the known good electronic document containing the good function call,

a code segment to add an entry for the intercepted good function call to the detection model, the entry including the at least one argument value, and

a code segment to repeat the intercepting and adding steps for each good function call in each known good electronic document, thereby building the detection model.

54. The non-transitory computer readable medium having stored thereupon computing instructions for detecting an anomaly in an electronic document of claim **53** where the code segment to add an entry for the intercepted good function call further includes a code segment to include in the entry a number of times the function call is intercepted by the detection engine while building the detection model.

55. The non-transitory computer readable medium of claim **54** further comprising:

a code segment to remove function call entries from the detection model where the number of times the function call was intercepted by the detection engine in processing the plurality of known good electronic documents is less than a threshold, after the plurality of known good electronic documents are processed by the application program.

* * * * *