An example system includes a processor to crawl a plurality of web pages of a web application to be tested. The processor is to also receive an intercepted input to the web application and an output from a web application associated with each crawled web page. The processor is to further detect testable elements in the intercepted input and the output. The processor is also to generate a fingerprint for each web page based on the detected testable elements. The processor is to generate a list of clusters comprising one or more similar web pages based on the fingerprints. The processor is to test a single web page from each cluster.
Crawl Plurality of Web Pages of a Web Application to be Tested

Receive Intercepted Input to Web Server Application and Output from Web Application Associated with Each Crawled Web Page

Detect Testable Elements in Intercepted Input and Output

Generate Fingerprint for Each Web Page Based on Detected Testable Elements

Generate List of Clusters Including One or More Similar Web Pages Based on Fingerprints

Test Single Web Page from Each Cluster

FIG. 2
Receive List of Clusters

Generate Maximal Distance for Each Cluster in List of Clusters

Receive Request to be Sent to Web Application

Request to be Sent Would Result in Web Page that Belongs to Cluster Based on Maximal Distances?

Send Request

Do Not Send Request

FIG. 3
FIG. 6
TESTING WEB APPLICATIONS USING CLUSTERS

BACKGROUND

[0001] The present techniques relate to testing web applications. More specifically, the techniques relate to testing web applications using clusters.

SUMMARY

[0002] According to an embodiment described herein, a system can include a processor to crawl a plurality of web pages of a web application to be tested. The processor can also further receive an intercepted input to the web application and an output from the web application associated with each crawled web page. The processor can also detect testable elements in the intercepted input and the output. The processor can also generate a fingerprint for each web page based on the detected testable elements. The processor can also generate a list of clusters comprising one or more similar web pages based on the fingerprints. The processor can also further test a single web page from each cluster.

[0003] According to another embodiment described herein, a method can include crawling, via a processor, a plurality of web pages of a web application to be tested. The method can also further include receiving, via the processor, an intercepted input to the web application and an output from the web application associated with each crawled web page. The method can also include detecting, via the processor, testable elements in the intercepted input and the output. The method can further include generating, via the processor, a fingerprint for each web page based on the detected testable elements. The method can also include generating, via the processor, a list of clusters comprising one or more similar web pages based on the fingerprints. The method can also further include testing, via the processor, a single web page from each cluster.

[0004] According to another embodiment described herein, a computer program product for testing web applications can include computer-readable storage medium having program code embodied therein. The computer-readable storage medium is not a transitory signal per se. The program code is executable by a processor to cause the processor to crawl a plurality of web pages of a web application to be tested. The program code can also cause the processor to receive an intercepted input to the web application and an output from the web application associated with each crawled web page. The program code can also cause the processor to detect testable elements in the intercepted input and the output. The program code can also cause the processor to generate a fingerprint for each web page based on the detected testable elements. The program code can also cause the processor to also further generate a list of clusters comprising one or more similar web pages based on the fingerprints. The program code can also cause the processor to test a single web page from each cluster.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWINGS

[0005] FIG. 1 is a block diagram of an example system that can test web applications using clusters.

[0006] FIG. 2 is an information flow diagram of an example method for testing applications pages using clusters.

[0007] FIG. 3 is a process flow diagram of an example method for testing web applications using cluster predictions.

[0008] FIG. 4 is a block diagram of an example cloud computing environment according to embodiments described herein.

[0009] FIG. 5 is an example abstraction model layers according to embodiments described herein; and

[0010] FIG. 6 is an example tangible, non-transitory computer-readable medium that can test web applications using clusters.

DETAILED DESCRIPTION

[0011] Dynamic web application testers can test web applications by crawling web applications in their entirety and testing all the elements of each web application. For example, dynamic testing of web applications may include two phases: crawling and testing. The objective of the crawling phase may be to identify all testable elements in a web application. For example, the testable element may include parameters, cookie values, etc. In the testing phase, the dynamic web application tester can then attack testable elements and validate tests on the testable elements. However, such dynamic web scanning may not be able to scan large web applications due to the vast number of web pages to crawl to and the amount of testable elements per page combined with the physical constraints of machines.

[0012] According to embodiments of the present techniques a processor may test web applications using clusters. For example, the processor can crawl a plurality of web pages of a web application to be tested. The processor may receive an intercepted input to a web application and an output from the web application associated with each crawled web page. For example, the input may include a Hypertext Transfer Protocol (HTTP) request and the output may include an HTTP response. The processor may also detect testable elements in the intercepted input and the output. The processor may further generate a fingerprint for each web page based on the detected testable elements. The processor may then generate a list of clusters comprising one or more similar web pages based on the fingerprints. The processor may then test a single web page from each cluster. For example, since each cluster may have been created by the same server-side application functionality, only one page from each cluster may be tested. Thus, the present techniques may allow to reduce the number of web pages to be tested and thus increase the efficiency of the dynamic web application tester. In some experiments, the number of web pages to be tested to provide the same testing coverage was shown to be reduced by a factor of 10-20x. The present techniques may be able to further increase efficiency by predicting which requests would result in additional web pages in a cluster that would be redundant for purposes of testing. For example, the processor may be able to generate a maximal distance between requests for each cluster in the list of clusters and detect that the request would not result in a web page that belongs to any cluster based on the maximal distances for the clusters. Thus, the techniques may enable additional efficiency for a web application tester once a list of clusters has been generated. Furthermore, in some examples, the processor can detect a security vulnerability based on the testing and modify the web application to prevent the security vulnerability. For example, the proces-
sor may remove characters from user input that can result in the execution of unauthorized scripts.

In some scenarios, the techniques described herein may be implemented in a cloud computing environment. As discussed in more detail below in reference to at least FIGS. 4, 5, and 6, a computing device configured to test web applications using clusters may be implemented in a cloud computing environment. It is understood in advance that although this disclosure may include a description on cloud computing, implementation of the teachings recited herein are not limited to a cloud computing environment. Rather, embodiments of the present invention are capable of being implemented in conjunction with any other type of computing environment now known or later developed.

Cloud computing is a model of service delivery for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, network bandwidth, servers, processing, memory, storage, applications, virtual machines, and services) that can be rapidly provisioned and released with minimal management effort or interaction with a provider of the service. This cloud model may include at least five characteristics, at least three service models, and at least four deployment models.

Characteristics are as follows:

On-demand self-service: a cloud consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with the service's provider.

Broad network access: capabilities are available over a network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, laptops, and PDAs).

Resource pooling: the provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to demand. There is a sense of location independence in that the consumer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter).

Rapid elasticity: capabilities can be rapidly and elastically provisioned, in some cases automatically, to quickly scale out and rapidly released to quickly scale in. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time.

Measured service: cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported providing transparency for both the provider and consumer of the utilized service.

Service Models are as follows:

Software as a Service (SaaS): the capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through a thin client interface such as a web browser (e.g., web-based email). The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings.

Platform as a Service (PaaS): the capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including networks, servers, operating systems, or storage, but has control over the deployed applications and possibly application hosting environment configurations.

Infrastructure as a Service (IaaS): the capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, deployed applications, and possibly limited control of select networking components (e.g., host firewalls).

Deployment Models are as follows:

Private cloud: the cloud infrastructure is operated solely for an organization. It may be managed by the organization or a third party and may exist on-premises or off-premises.

Community cloud: the cloud infrastructure is shared by several organizations and supports a specific community that has shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be managed by the organizations or a third party and may exist on-premises or off-premises.

Public cloud: the cloud infrastructure is made available to the general public or a large industry group and is owned by an organization selling cloud services.

Hybrid cloud: the cloud infrastructure is composed of two or more clouds (private, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load-balancing between clouds).

A cloud computing environment is service oriented with a focus on statelessness, low coupling, modularity, and semantic interoperability. At the heart of cloud computing is an infrastructure comprising a network of interconnected nodes.

With reference now to FIG. 1, an example computing device can test web applications using clusters. The computing device 100 may be for example, a server, a network device, desktop computer, laptop computer, tablet computer, or smartphone. In some examples, computing device 100 may be a cloud computing node. Computing device 100 may be described in the general context of computer system executable instructions, such as program modules, being executed by a computer system. Generally, program modules may include routines, programs, objects, components, logic, data structures, and so on that perform particular tasks or implement particular abstract data types. Computing device 100 may be described in the general context of computer system executable instructions, such as program modules, being executed by a computer system. Generally, program modules may include routines, programs, objects, components, logic, data structures, and so on that perform particular tasks or implement particular abstract data types.
environment, program modules may be located in both local and remote computer system storage media including memory storage devices.

[0032] The computing device 100 may include a processor 102 that is to execute stored instructions, a memory device 104 to provide temporary memory space for operations of said instructions during operation. The processor can be a single-core processor, multi-core processor, computing cluster, or any number of other configurations. The memory 104 can include random access memory (RAM), read only memory, flash memory, or any other suitable memory systems.

[0033] The processor 102 may be connected through a system interconnect 106 (e.g., PCI®, PCI-Express®, etc.) to an input/output (I/O) device interface 108 adapted to connect the computing device 100 to one or more I/O devices 110. The I/O devices 110 may include, for example, a keyboard and a pointing device, wherein the pointing device may include a touchpad or a touchscreen, among others. The I/O devices 110 may be built-in components of the computing device 100, or may be devices that are externally connected to the computing device 100.

[0034] The processor 102 may also be linked through the system interconnect 106 to a display interface 112 adapted to connect the computing device 100 to a display device 114. The display device 114 may include a display screen that is a built-in component of the computing device 100. The display device 114 may also include a computer monitor, television, or projector, among others, that is externally connected to the computing device 100. In addition, a network interface controller (NIC) 116 may be adapted to connect the computing device 100 through the system interconnect 106 to the network 118. In some embodiments, the NIC 116 can transmit data using any suitable interface or protocol, such as the internet small computer system interface, among others. The network 118 may be a cellular network, a radio network, a wide area network (WAN), a local area network (LAN), or the internet, among others. An external computing device 120 may connect to the computing device 100 through the network 118. In some examples, external computing device 120 may be an external web-server 120. In some examples, external computing device 120 may be a computing node.

[0035] The processor 102 may also be linked through the system interconnect 106 to a storage device 122 that can include a hard drive, an optical drive, a USB flash drive, an array of drives, or any combinations thereof. In some examples, the storage device may include a crawler module 124, a receiver module 126, a detector module 128, a fingerprint generator module 130, a cluster generator module 132, a tester module 134, and a predictor module 136. In some examples, one or more of the modules 124-136 may be implemented in an application or a web browser plugin. The crawler module 124 can crawl a plurality of web pages of a web application to be tested. For example, given one or more seed uniform resource locators (URLs), the crawler module 124 can download the web pages associated with the URLs, extract any hyperlinks contained in the URLs, and add the hyperlinks to a list of URLs to visit, also known as a crawl frontier.

[0036] URLs from the crawl frontier are then recursively visited according to the crawler's policy. The receiver module 126 can then receive an intercepted input to a web application and an output from the web application associated with each crawled web page. For example, the input may include an HTTP request and the output may include an HTTP response. In some examples, the input may include a GET parameter and the output may include a document object model. For example, the document object model may be a tree structure of a web page. The detector module 128 can detect testable elements in the intercepted input and the output. The fingerprint generator module 130 can generate a fingerprint for each web page based on the detected testable elements. For example, the fingerprint for each web page may include a plurality of response element fingerprints and a plurality of request element fingerprints. For example, the response element fingerprints may include concatenated fingerprints of a number of extracted response elements in a response. The request element fingerprints may include concatenated fingerprints of a number of extracted request elements in a request. In some examples, the fingerprint generator 130 can calculate a similarity score between the fingerprint for each web page and each cluster in the list of clusters based on a calculated hamming distance. The cluster generator module 132 can generate a list of clusters comprising one or more similar web pages based on the fingerprints. For example, the cluster generator 132 can calculate a similarity score between a fingerprint for a web page from the plurality of web pages and a cluster from the list of clusters. The cluster generator 132 can then add the web page to the cluster in response to detecting that the similarity score exceeds a similarity threshold. In some examples, the cluster generator module 132 can calculate a similarity score between the fingerprint for each web page and each cluster in the list of clusters based on a calculated hamming distance or any other suitable linear block code technique. In some examples, the cluster generator module 132 can calculate a hamming distance for a fingerprint of a web page and a cluster by detecting the number of positions in two compared fingerprints at which corresponding symbols are different. For example, the hamming distance may indicate the number of substitutions to be made to change one fingerprint into the other. The tester module 134 can test a single web page from each cluster. Thus, the tester module 134 may efficiently test web applications by testing a single web page from each cluster and not every web page, while maintaining testing coverage.

[0037] In some examples, the predictor module 136 can generate a maximal distance between requests for each cluster in the list of clusters. In some examples, the predictor module 136 may receive a request to be sent to the web application. The predictor module 136 can then detect that the request would not result in a web page that belongs to any cluster based on the maximal distances for the clusters. In some examples, the crawler module 124 can then send the request to a web application in response to detecting that the request would not result in the web page that belongs to any cluster. In some examples, the crawler module 124 may not send the request to a web application in response to detecting that the request would result in the web page that belongs to a cluster. Thus, the predictor module 136 may enable the crawler module 124 to perform more efficiently by sending a reduced number of requests and enable the receiver module 126 to perform more efficiently by not receiving web pages that would not be used in testing.

[0038] It is to be understood that the block diagram of FIG. 1 is not intended to indicate that the computing device 100 is to include all of the components shown in FIG. 1.
Rather, the computing device 100 can include fewer or additional components not illustrated in FIG. 1 (e.g., additional memory components, embedded controllers, modules, additional network interfaces, etc.). Furthermore, any of the functionalities of the crawler module 124, the receiver module 126, the detector module 128, the fingerprint generator module 130, the cluster generator module 132, the tester module 134, and the predictor module 136, may be partially, or entirely, implemented in hardware and/or in the processor 102. For example, the functionality may be implemented with an application specific integrated circuit, logic implemented in an embedded controller, or in logic implemented in the processor 102, among others. In some embodiments, the functionalities of the crawler module 124, the receiver module 126, the detector module 128, the fingerprint generator module 130, the cluster generator module 132, the tester module 134, and the predictor module 136, can be implemented with logic, wherein the logic, as referred to herein, can include any suitable hardware (e.g., a processor, among others), software (e.g., an application, among others), firmware, or any suitable combination of hardware, software, and firmware.

FIG. 2 is a process flow diagram of an example method for testing applications pages using clusters. The method 200 can be implemented with any suitable computing device, such as the computing device 100 of FIG. 1. For example, the method can be implemented via the processor 102 of computing device 100.

At block 202, a processor crawls a plurality of web pages of a web application to be tested. For example, the web pages of the web application may be recursively visited according to a crawling policy. In some examples, the crawling policy may be a selection policy that states the pages to be downloaded. For example, a selection policy may be a focused crawling policy that looks for similarity of web pages to a given query. In some examples, a selection policy may be a URL normalization policy that tries to normalize a URL to avoid redundant crawling. In some examples, the crawling policy may be a parallelization policy that uses many crawlers in parallel to increase crawling efficiency. For example, the parallelization policy can state how to coordinate a plurality of distributed web crawlers.

At block 204, the processor receives an intercepted input to the web application and an output from a web application associated with each crawled web page. For example, the intercepted input may be a HTTP request. In some examples, the output may be an HTTP response. In some examples, the processor can intercept input to a web application, send the input to the web application if the input would not result in a web page that belongs to any cluster based on the maximal distances calculated for list of clusters, and receive an output from the web application in response to the input.

At block 206, the processor detects testable elements in the intercepted input and the output. For example, testable elements in the input can include schemes, ports, parameters, cookies, headers, etc. In some examples, the testable elements in the output can include document object model (DOM) elements of HTTP responses.

At block 208, the processor generates a fingerprint for each web page based on the detected testable elements. For example, the processor can generate a fingerprint for each element in a request resulting in the web page and a fingerprint for each element in the web page and combine the fingerprints for the elements to generate the fingerprint for each web page. In some examples, the fingerprint for a web page may be generated using a dictionary or vector that counts the number of occurrences of testable elements in the requests and responses of a web page. For example, the processor may count the number of query or body parameters in the request. In some examples, the processor can count the number of specific HTML elements in the response. Thus, the fingerprint may be based on a concatenation of both the fingerprints generated for input elements and fingerprints generated for output elements.

At block 210, the processor generates a list of clusters comprising one or more similar web pages based on the fingerprints. For example, the clusters may represent web pages with similar server-side functionality. In some examples, the web pages in a cluster may be the output of the same server-side functionality or script. In some examples, the processor may take two of the fingerprints and calculate a similarity based on an average of the ratios of each of the testable elements. In some examples, the processor can calculate a similarity score between a fingerprint for a web page from the plurality of web pages and a cluster from the list of clusters and add the web page to the cluster in response to detecting that the similarity score exceeds a similarity threshold. In some examples, the processor can calculate a similarity score between the fingerprint for each web page and each cluster in the list of clusters based on a calculated hamming distance. For example, the hamming distance may indicate number of positions in two compared fingerprints at which corresponding symbols are different. In some examples, the processor can calculate a similarity score by calculating min-wise independent permutations (MinHash). For example, MinHash may be used to calculate the ratio of the number of elements of an intersection between two fingerprints and the number of elements of their union without explicitly computing the intersection and the union. In some examples, the processor may calculate a similarity score using a SimHash hashing function. For example, similar elements may be hashed to similar hash values having low hamming distances.

At block 212, the processor tests a single web page from each cluster. For example, since each cluster is believed to have been created by the same server side functionality, only one page from each cluster may be tested. In some examples, the processor can then detect a security vulnerability based on the testing and modify the web application to prevent the security vulnerability. For example, the processor may remove characters from user input that can result in the execution of unauthorized scripts. In some examples, if a security vulnerability is detected from the single page tested, the processor can modify the web application to affect each of the pages of the cluster and prevent the security vulnerability.

The process flow diagram of FIG. 2 is not intended to indicate that the operations of the method 200 are to be executed in any particular order, or that all of the operations of the method 200 are to be included in every case. Additionally, the method 200 can include any suitable number of additional operations.

FIG. 3 is a process flow diagram of an example method for testing web applications using cluster predictions. The method 300 can be implemented with any suitable computing device, such as the computing device 100 of FIG.
For example, the method can be implemented via the processor 102 of computing device 100.

At block 302, the processor receives a list of clusters. For example, the list of clusters may have been generated using the method 200 above.

At block 304, the processor generates a maximal distance for each cluster in the list of clusters. For example, the processor can generate a maximal distance between requests for a cluster by calculating similarity scores between requests resulting in the web pages in the cluster. The maximal distance may be a similarity score that is lower than other similarity scores in the cluster. In some examples, the similarity scores can be calculated using any suitable form of hamming distance. In some examples, the maximal distance between requests in a cluster can be stored as a variable of the cluster. Thus, for every web-page that is inserted into the cluster, the processor can consider the request that preceded the web page, and iteratively compute the similarity between the request and other requests in the cluster. For example, a similarity score can be calculated using any suitable variation of a hamming distance. In some examples, the hamming distance can be performed on a portion of the request, rather than every part of the request.

At block 306, the processor receives a request to be sent to a web application. For example, the request to be sent may be a textual representation of a request that may have been intercepted before reaching the web application. In some examples, the request may be sent to the target web application in a test environment. For example, the request to be sent may be generated from a received test input. In some examples, the request can be generated automatically. For example, the processor may generate the request by parsing previous responses and extracting new links. In some examples, the request can be given directly. For example, the processor may receive starting URLs, also known as seeds, and generate the request based on the received starting URLs. In some examples, the request can be generated by intercepting data from a client to the web application. For example, processor may intercept the data at a browser.

At decision diamond 308, the processor determines whether the request to be sent would result in a web page that belongs to a cluster based on the maximal distances. For example, if the request is within the maximal distance of a cluster then the processor may detect that the request would result in a web page that would belong to the cluster. If the request would not result in a web page that belongs to a cluster, then the method may continue at block 310. If the request would result in a web page that belongs to a cluster, then the method may continue at block 312.

At block 310, the processor sends the request. For example, the processor may send the request to a target web application. For example, the requests may be sent to the web application in a testing environment. The processor may then receive a response that can be processed according to the method 200 of FIG. 2 above.

At block 312, the processor does not send the request. For example, since the request may not return any useful web page for testing, both the resources used to send the request and the resources used to receive the web page can be used for other processing. Thus, the processor may be able to more efficiently test the web application by reducing the number of web pages to be clustered.
access to the cloud computing environment for consumers and system administrators. Service level management provides cloud computing resource allocation and management such that required service levels are met. Service Level Agreement (SLA) planning and fulfillment provide pre-arrangement for, and procurement of, cloud computing resources for which a future requirement is anticipated in accordance with an SLA.

[0059] Workloads layer 506 provides examples of functionality for which the cloud computing environment may be utilized. Examples of workloads and functions which may be provided from this layer include: mapping and navigation; software development and lifecycle management; virtual classroom education delivery; data analytics processing; transaction processing; and web application testing.

[0060] The present techniques may be a system, a method or computer program product. The computer program product may include a computer readable storage medium (or media) having computer readable program instructions thereon for causing a processor to carry out aspects of the present invention.

[0061] The computer readable storage medium can be a tangible device that can retain and store instructions by use for an instruction execution device. The computer readable storage medium may be, for example, but is not limited to, an electronic storage device, a magnetic storage device, an optical storage device, an electromagnetic storage device, a semiconductor storage device, or any suitable combination of the foregoing. A non-exhaustive list of more specific examples of the computer readable storage medium includes the following: a portable computer diskette, a hard disk, a random access memory (RAM), a read-only memory (ROM), an erasable programmable read-only memory (EPROM or Flash memory), a static random access memory (SRAM), a portable compact disc read-only memory (CD-ROM), a digital versatile disk (DVD), a memory stick, a floppy disk, a mechanically encoded device such as punch-cards or raised structures in a groove having instructions recorded thereon, and any suitable combination of the foregoing. A computer readable storage medium, as used herein, is not to be construed as being transitory signals per se, such as radio waves or other freely propagating electromagnetic waves, electromagnetic waves propagating through a waveguide or other transmission media (e.g., light pulses passing through a fiber-optic cable), or electrical signals transmitted through a wire.

[0062] Computer readable program instructions described herein can be downloaded to respective computing/processing devices from a computer readable storage medium or to an external computer or external storage device via a network. For example, the Internet, a local area network, a wide area network and/or a wireless network. The network may comprise copper transmission cables, optical transmission fibers, wireless transmission, routers, firewalls, switches, gateway computers and/or edge servers. A network adapter card or network interface in each computing/processing device receives computer readable program instructions from the network and forwards the computer readable program instructions for storage in a computer readable storage medium within the respective computing/processing device.

[0063] Computer readable program instructions for carrying out operations of the present techniques may be assembler instructions, instruction-set-architecture (ISA) instructions, machine instructions, machine dependent instructions, microcode, firmware instructions, state-setting data, or either code or object code written in any combination of one or more programming languages, including an object oriented programming language such as Smalltalk, C++ or the like, and conventional procedural programming languages, such as the "C" programming language or similar programming languages. The computer readable program instructions may execute entirely on the user’s computer, partly on the user’s computer, as a stand-alone software package, partly on the user’s computer and partly on a remote computer or entirely on the remote computer or server. In the latter scenario, the remote computer may be connected to the user’s computer through any type of network, including a local area network (LAN) or a wide area network (WAN), or the connection may be made to an external computer (for example, through the Internet using an Internet Service Provider). In some embodiments, electronic circuitry including, for example, programmable logic circuitry, field-programmable gate arrays (FPGA), or programmable logic arrays (PLA) may execute the computer readable program instructions by utilizing state information of the computer readable program instructions to personalize the electronic circuitry, in order to perform aspects of the present techniques.

[0064] Aspects of the present techniques are described herein with reference to flowchart illustrations and/or block diagrams of methods, apparatus (systems), and computer program products according to embodiments of the techniques. It will be understood that each block of the flowchart illustrations and/or block diagrams, and combinations of blocks in the flowchart illustrations and/or block diagrams, can be implemented by computer readable program instructions.

[0065] These computer readable program instructions may be provided to a processor of a general purpose computer, special purpose computer, or other programmable data processing apparatus to produce a machine, such that the instructions, which execute via the processor of the computer or other programmable data processing apparatus, create means for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks. These computer readable program instructions may also be stored in a computer readable storage medium that can direct a computer, a programmable data processing apparatus, and/or other devices to function in a particular manner, such that the computer readable storage medium having instructions stored therein comprises an article of manufacture including instructions which implement aspects of the function/act specified in the flowchart and/or block diagram block or blocks.

[0066] The computer readable program instructions may also be loaded onto a computer, other programmable data processing apparatus, or other device to cause a series of operational steps to be performed on the computer, other programmable apparatus or other device to produce a computer implemented process, such that the instructions which execute on the computer, other programmable apparatus, or other device implement the functions/acts specified in the flowchart and/or block diagram block or blocks.

[0067] Referring now to FIG. 6, a block diagram is depicted of an example tangible, non-transitory computer-readable medium 600 that can test web applications using clusters. The tangible, non-transitory, computer-readable
medium 600 may be accessed by a processor 602 over a computer interconnect 604. Furthermore, the tangible, non-transitory, computer-readable medium 600 may include code to direct the processor 602 to perform the operations of the methods 200 and 300 of FIGS. 2 and 3 above.

[0068] The various software components discussed herein may be stored on the tangible, non-transitory, computer-readable medium 600, as indicated in FIG. 6. For example, a crawler module 606 includes code to crawl a plurality of web pages of a web application to be tested. A receiver module 608 includes code to receive an intercepted input to the web application and an output from a web application associated with each crawled web page. For example, the input may include an HTTP request or a GET parameter and the output may include an HTTP response or a document object model. A detector module 610 includes code to detect testable elements in the intercepted input and the output. A fingerprint generator module 612 includes code to generate a fingerprint for each web page based on the detected testable elements. For example, the fingerprint generator module 612 can include code to generate a fingerprint for each element in a request resulting in the web page and a fingerprint for each element in the web page. The fingerprint generator module 612 also include code to combine the fingerprints for the elements to generate the fingerprint for each web page. A cluster generator module 614 includes code to generate a list of clusters including one or more similar web pages based on the fingerprints. For example, the cluster generator module 614 can include code to calculate a similarity score between a fingerprint for a web page from the plurality of web pages and a cluster from the list of clusters and add the web page to the cluster in response to detecting that the similarity score exceeds a similarity threshold. In some examples, the cluster generator module 614 can include code to calculate a similarity score between the fingerprint for each web page and each cluster in the list of clusters based on a calculated hamming distance. A tester module 616 includes code to test a single web page from each cluster. A predictor module 618 includes code to generate a maximal distance between requests for each cluster in the list of clusters. In some examples, the predictor module 618 can include code to generate a maximal distance between requests for a cluster by calculating similarity scores between requests resulting in the web pages in the cluster. For example, the maximal distance may be a similarity score that is lower than the other similarity scores. The predictor module 618 can also include code to receive a request to be sent to the web application. The predictor module 618 can include code to detect that the request would not result in a web page that belongs to any cluster based on the maximal distances for the clusters. The crawler module 606 can then send the request in response to detecting that the request would not result in the web page that belongs to any cluster. Thus, the predictor module 618 may enable the crawler module 606 to operate more efficiently by reducing unnecessary requests from being sent and also prevent unnecessary web pages from being received at the receiver module 608. It is to be understood that any number of additional software components not shown in FIG. 6 may be included within the tangible, non-transitory, computer-readable medium 600, depending on the particular application.

[0069] The flowchart and block diagrams in the Figures illustrate the architecture, functionality, and operation of possible implementations of systems, methods, and computer program products according to various embodiments of the present techniques. In this regard, each block in the flowchart or block diagrams may represent a module, segment, or portion of instructions, which comprises one or more executable instructions for implementing the specified logical function(s). In some alternative implementations, the functions noted in the block may occur out of the order noted in the figures. For example, two blocks shown in succession may, in fact, be executed substantially concurrently, or the blocks may sometimes be executed in the reverse order, depending upon the functionality involved. It will also be noted that each block of the block diagrams and/or flowchart illustration, and combinations of blocks in the block diagrams and/or flowchart illustration, can be implemented by special purpose hardware-based systems that perform the specified functions or acts or carry out combinations of special purpose hardware and computer instructions.

[0070] The descriptions of the various embodiments of the present techniques have been presented for purposes of illustration, but are not intended to be exhaustive or limited to the embodiments disclosed. Many modifications and variations will be apparent to those of ordinary skill in the art without departing from the scope and spirit of the described embodiments. The terminology used herein was chosen to best explain the principles of the embodiments, the practical application or technical improvement over technologies found in the marketplace, or to enable others of ordinary skill in the art to understand the embodiments disclosed herein.

What is claimed is:

1. A computer system for testing web applications, the computer system comprising:
   - one or more computer processor;
   - one or more computer readable storage media; and
   - program instructions, stored on the one or more computer readable storage media for execution by at least one of the one or more computer processors, the program instructions comprising:
     - program instructions to receive an intercepted input to a web application;
     - program instructions to cause the intercepted input to be sent to the web application as a request input where a similarity score between the intercepted input and any one of the one or more web page requests is less than a maximal distance score of the given web cluster, wherein the similarity score is between the intercepted input and one or more web page requests, where each web page request is associated with a web page of a given web cluster;
     - program instructions to receive the output for the intercepted input from the web application;
     - program instructions to detect testable elements in the intercepted input and output;
     - program instructions to generate a combined fingerprint for the intercepted input and output based on the detected testable elements from the intercepted input and the output by counting the number of occurrences of the detected testable elements in the intercepted input and the output;
     - program instructions to add the output to one cluster of the list of web page clusters based on similarity between the combined fingerprint and the one cluster; and
program instructions to test a single web page from each cluster of the list of web page clusters.

2. The computer system of claim 1, wherein the intercepted input comprises a hypertext transfer protocol (HTTP) request and the output comprises a hypertext transfer protocol (HTTP) response.

3. The computer system of claim 1, wherein the test of the single web page from the given web cluster includes the processor being further configured to detect a security vulnerability based on the intercepted input; and modify the web application to prevent the detected security vulnerability in each web page of the given web cluster.

4. The computer system of claim 1, wherein the processor is configured to calculate a similarity score between a combined fingerprint for a respective web page from the web cluster and a respective cluster from a list of clusters and add the respective web page to the respective cluster in response to detecting that the similarity score exceeds a similarity threshold.

5. The computer system of claim 1, wherein the processor is configured to calculate a similarity score between the combined fingerprint and each cluster in a list of clusters based on a calculated hamming distance.

6. The computer system of claim 1, wherein the combined fingerprint for each web page comprises a plurality of response element fingerprints and a plurality of request element fingerprints.

7. The computer system of claim 1, wherein the maximal distance score of the given web cluster is generated by calculating the similarity of each of one or more web page requests that resulted in a web page associated with the given web page cluster.

8. A computer-implemented method for testing web applications, the method comprising the steps of:

   receiving, by one or more computer processors, an intercepted input to a web application where the intercepted input is configured to cause the web application to provide a responsive web page as an output for the intercepted input;

   causing, by one or more computer processors, the intercepted input to be sent to the web application as a request input where a similarity score between the intercepted input and any one of the one or more web page requests is less than a maximal distance score of the given web cluster, wherein the similarity score is between the intercepted input and one or more web page requests, where each web page request is associated with a web page of a given web cluster;

   receiving, by one or more computer processors, the output for the intercepted input from the web application,

   detecting, by one or more computer processors, testable elements in the received request and response;

   generating, via a processor, a combined fingerprint for each web page based on a first fingerprint generated from the detected testable elements from the intercepted request and a second fingerprint generated from the detected testable elements of the response by counting the number of occurrences of the detected testable elements in the intercepted input and the output;

   adding, by one or more computer processors, the output to one cluster of the list of web page clusters based on similarity between the combined fingerprint and the one cluster;

   testing, by one or more computer processors, a single web page from each cluster of the list of web page clusters.

9. The computer-implemented method of claim 8, further comprising:

   calculating, by one or more computer processors, a similarity score between the combined fingerprint for a web page from the plurality of web pages and a cluster from a list of clusters; and

   adding, by one or more computer processors, the web page to the cluster in response to detecting that the similarity score exceeds a similarity threshold.

10. The computer-implemented method of claim 8, further comprising calculating, by one or more computer processors, a similarity score between the fingerprint and each cluster in a list of clusters based on a calculated hamming distance.

11. The computer-implemented method of claim 8, wherein the step of generating the combined fingerprint for each web page comprises the steps of:

   generating, by one or more computer processors, a first fingerprint representing each element in a GET request and a second fingerprint for each element of a document object model returned as a response to the GET request; and

   combining, by one or more computer processors, the first and second fingerprints for the elements to generate the combined fingerprint for each web page.

12. The computer-implemented method of claim 8, further comprising steps of:

   sending, by one or more computer processors, the intercepted input to the web application if the intercepted input would not result in a web page that belongs to any cluster based on maximal distances calculated for the list of clusters; and

   receiving, by one or more computer processors, an output from the web application in response to the intercepted input.

13. The computer-implemented method of claim 8, further comprising the step of generating, by one or more computer processors, the a maximal distance between requests for a cluster by calculating similarity scores between requests resulting in the web pages in the given web cluster, wherein the maximal distance comprises a similarity score that is lower than other similarity scores.

14. The computer-implemented method of claim 8, wherein the maximal distance score of the given web cluster is generated by calculating the similarity of each of the one or more web page requests that resulted in a web page associated with the given web page cluster.

15. A computer program product for testing web applications, the computer program product comprising:

   one or more computer readable storage media; and

   program instructions stored on the one or more computer readable storage media, the program instructions comprising:

   program instructions to receive an intercepted input to a web application, where the intercepted input is configured to cause the web application to provide a responsive web page as an output for the intercepted input;

   program instructions to cause the intercepted input to be sent to the web application as a request input where a similarity score between the intercepted input and any one of the one or more web page
requests is less than a maximal distance score of the given web cluster, wherein the similarity score is between the intercepted input and one or more web page requests, where each web page request is associated with a web page of a given web cluster; program instructions to receive the output for the intercepted input from the web application; program instructions to detect testable elements in the intercepted input and the output; program instructions to generate a combined fingerprint for each web page based on a first fingerprint of the detected testable elements from the intercepted input and a second fingerprint of the detected testable element from the output by counting the number of occurrences of the detected testable elements in the intercepted input and the output; program instructions to add the output to one cluster of the list of web page clusters comprising one or more similar web pages based on similarity between the combined fingerprint and the one cluster; and program instructions to test a single web page from each cluster of the list of web page clusters.

16. The computer program product of claim 15, further comprising program instructions, stored on the one or more computer readable storage media, to calculate a similarity score between a fingerprint for a web page from the plurality of web pages and a cluster from a list of clusters, and to add the web page to the cluster in response to detecting that the similarity score exceeds a similarity threshold.

17. The computer program product of claim 15, further comprising program instructions, stored on the one or more computer readable storage media, to calculate a similarity score between the fingerprint for each web page and each cluster in a list of clusters based on a calculated hamming distance.

18. The computer program product of claim 15, further comprising program instructions, stored on the one or more computer readable storage media, to:
   - generate a fingerprint for each element in a request resulting in the web page and a fingerprint for each element in the web page; and
   - combine the fingerprints for the elements to generate the fingerprint for each web page.

19. The computer program product of claim 15, further comprising program instructions, stored on the one or more computer readable storage media, to generate a maximal distance between requests for a cluster by calculating similarity scores between requests resulting in the web pages in the cluster, wherein the maximal distance comprises a similarity score that is lower than other similarity scores.

20. The computer program product of claim 15, wherein the maximal distance score of the given web cluster is generated by calculating the similarity of each of the one or more web page requests that resulted in a web page associated with the given web page cluster.