RC25662 (WAT1709-054) September 18, 2017
Computer Science

IBM Research Report

A Cloud-Based Service That Protects End-User Devices from Malware in Email Attachments and Web Links

Anrin Chakraborti, Rick Boivie, Zhongshu Gu, Mehmet Kayaalp,
Ankita Lamba, Dimitrios Pendarakis
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598 USA

Research Division
Almaden – Austin – Beijing – Brazil – Cambridge – Dublin – Haifa – India – Kenya – Melbourne – T.J. Watson – Tokyo – Zurich

A Cloud-based Service that Protects End-User Devices from Malware in Email Attachments and Web Links

Abstract:

We have seen a significant increase in cyberattacks that leverage malware-bearing email attachments and malware-infected web sites. A recent report by Symantec[1] reports that 1 in every 359 emails sent in July, 2017 included malware – a 20% increase over previous months. Even more alarming is the fact that such malware is inexpensive and readily available for purchase[2].

The root of this problem is the lack of a mechanism that allows users to open email attachments and visit web sites *safely*. Today, when a user clicks on an attachment in an email, the user's software opens the attachment with a program such as Adobe Reader or in a browser tab and the user's device can become infected if the program has a vulnerability that an attacker can exploit with a carefully designed attachment. Similarly, when a user clicks on a link to a web site, the user's device can become infected if the web site contains malware. Unfortunately, existing solutions such as anti-virus software are not foolproof and are vulnerable to previously unknown (zero-day) attacks.

We propose a "lightweight" Cloud-based Service that can protect a user's "device" (which can be a laptop computer, or a mobile device such as an iphone, ipad or android device) from malware in email attachments and web sites without adversely affecting the user experience. By protecting the user device, the Service also prevents the malware from establishing a "beachhead" on a device that could be used to infect other systems in a business or other enterprise. The Cloud-based Service leverages 1) a Secure CPU technology that protects the confidentiality and integrity of a "Secure Object" from the other software on a system, 2) virtualization technology that is used in conjunction with the Secure CPU technology to provide "Secure Virtual Machines", and 3) a graphical desktop sharing tool that allows a user to safely interact with an attachment or a web site through a secure virtual machine.

The Cloud-based Service leverages an extension to a web browser (Google Chrome in our proof of concept implementation) and provides several protections: 1) it protects the integrity of client devices and enterprises from the unintentional downloading of malware when a user opens an attachment or clicks on a web link; 2) It protects the confidentiality of user information by protecting the integrity of client devices and by protecting client information within secure virtual machines; 3) it protects the integrity of any public keys or digital certificates that a secure virtual machine may use to authenticate the identity of web sites (e.g. so that a user can have a high-level of confidence that he is connected to his bank's web site, say, and not a fraudulent web site that has been set up to collect credentials and other information).

Importantly, the Cloud-based Service can protect against these attacks -- including previously unknown (zero-day) attacks -- without having to determine whether an email attachment or web site is malicious. The Cloud-based Service doesn't know and doesn't care.

[1] https://www.symantec.com/connect/blogs/latest-intelligence-july-2017

[2] https://www.forbes.com/sites/leemathews/2017/07/17/new-password-stealing-malware-spreads-rapidly-thanks-to-rock-bottom-pricing/#27eb41b36f16

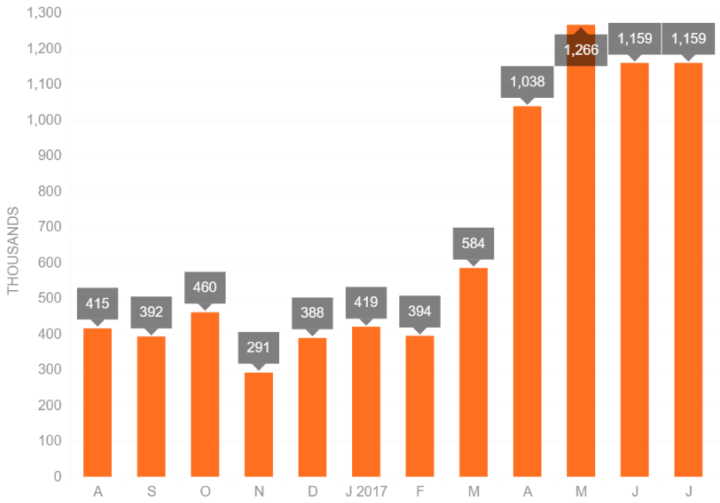

Figure 1. Number of web attacks per day in the last year[1]. Currently the number of web attacks per day stands at 1,158,985. The number has almost doubled from March 2017 to April 2017.

	Name*	% of all attacks**
1	Malicious URL	77.26
2	Trojan-Clicker.HTML.Iframe.dg	8.15
3	Trojan.Script.Generic	6.74
4	Trojan.Script.Iframer	3.14
5	Trojan-Downloader.Script.Generic	0.35
6	Exploit.Script.Generic	0.20
7	Packed.Multi.MultiPacked.gen	0.15
8	Trojan.JS.FBook.bh	0.13
9	Exploit.Script.Blocker	0.11
10	Trojan-Downloader.JS.Iframe.div	0.11

Figure 2. Sources of web attacks and their percentage contribution to the total number of web attacks overall. More than 77% of web attacks are through malicious URLs.

I. Introduction

With the web becoming increasingly more complex every day, the avenues for undermining the security of devices connected to the web increases exponentially. In general, these attacks try to exploit vulnerabilities in software running on the connected devices (mobiles, laptops, tablets etc). Since systems today run a wide range of complex software, it is extremely difficult to prevent attacks by detecting and patching vulnerabilities in all this software. It is estimated

that currently the number of web attacks per day stands at 1,158,985 (Figure 1). As shown by a recent report[3] (Figure 2), more than 77% of these attacks are due to malicious URLs.

Malicious URLs direct users to web sites that can download malware to a user's device. The malware can exploit other vulnerabilities in a system and it can even acquire administrator privileges. Although, whitelisting/blacklisting based techniques can prevent users from visiting some malicious URLs, these methods are far from foolproof.

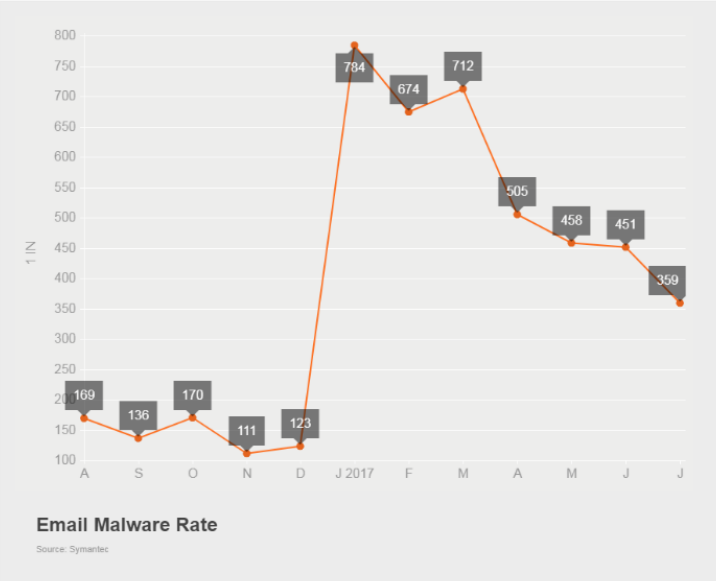

Figure 3. Number of Emails infected with Malware over last one year

Another commonly used attack vector is through email attachments. In this case, the user receives an email from a supposedly trusted entity with an attachment or web link. The user then downloads the attachment, which may be malware, to his or her local device. Figure 3 depicts the number of emails sent/received in the last year which contained some form of malware. Unfortunately, this rate has been increasing with the month of July 2017 witnessing a 20% increase over previous months. Currently, 1 out of every 359 emails sent contains malware. Two very recent examples of malware spread through email attachments are the OvidyStealer[4] and the Petya Ransomware[5].

At the root of this problem is the lack of a mechanism that allows users to process email attachments and web links safely. When a user clicks on an attachment in an email, either the email program opens the attachment in a program such as Adobe Reader or in modern browsers (such as Google Chrome), the attachment is opened in a new tab in the browser. For security purposes, modern browsers ensure that each new tab is running in a sandboxed environment to protect other software from the malware that might be running in a browser

tab. Unfortunately, if the browser has vulnerabilities, malware can escape the sandbox and infect other software. Similarly, vulnerabilities in other software that processes attachments can be exploited with a carefully designed attachment.

Existing solutions are “far from perfect”. Blacklists, whitelists and anti-malware software are prone to “zero-day” attacks. That is, these measures fail against unknown attacks. Since, new malware is being designed rapidly and with sophisticated techniques to evade detection by existing software/lists, it is imperative to design a solution that does not rely on solving the non-trivial problem of successfully detecting malware.

In this work, we propose a different approach to safely handle email attachments and web links. To protect against zero-day attacks, our method is agnostic to the specific type/design of malware or its source. Instead the user can browse web links and process attachments (even if they are infected) with the assurance that the malware will not be downloaded to the user’s device. To this end, we propose a cloud-based service that allows users to browse links/attachments in a “safe remote environment” which is discarded after use. This ensures that any malware that may have been present in the links/attachments does not infect the user’s device. The cloud-based service leverages a Secure CPU technology to protect the confidentiality and integrity of a "Secure Object" from the other software on a system, virtualization technology that is used in conjunction with the Secure CPU technology to provide "secure virtual machines", and a graphical desktop sharing tool that allows a user to safely interact with an attachment or a web site through a secure virtual machine.

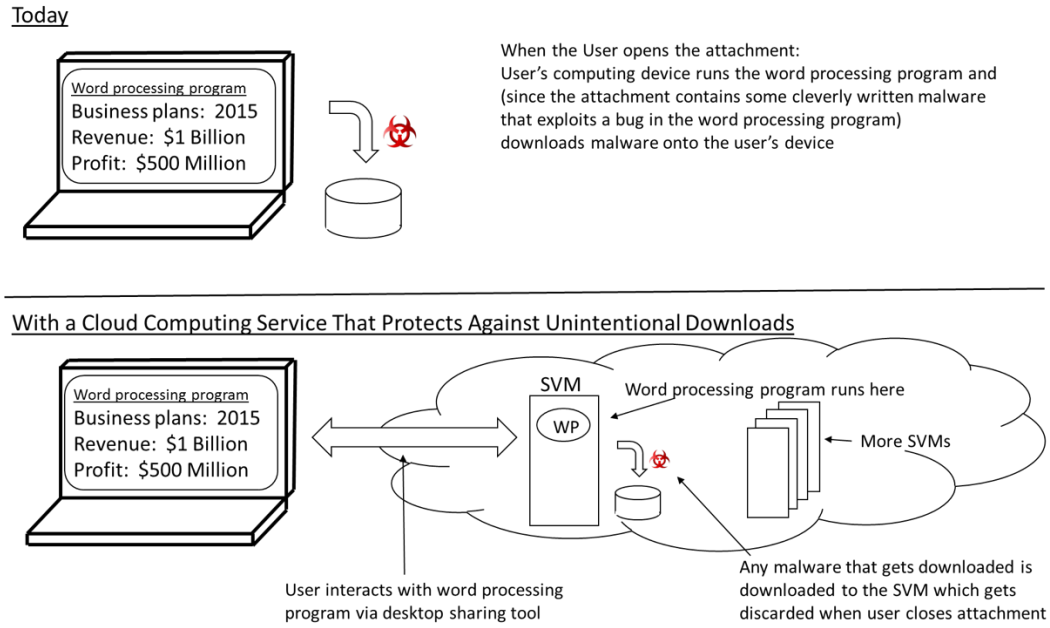

Figure 4. The cloud service proposed here vs. the current mechanism to process attachments in mail and their security implications.

II. Secure Virtual Machine

In previous work[6] we defined the notion of a 'Secure Object' comprising code and data that is cryptographically protected so that the other software on a computer system cannot access or undetectably tamper with the information in a Secure Object and we described technology for supporting these Secure Objects. The technology protects the confidentiality and integrity of information in a Secure Object from other software while making this information available to the Secure Object itself. Secure Objects can be used to protect individual applications from the other software on a system including privileged software like an operating system, device drivers and applications that run with root privileges -- and malware that obtains root privilege by exploiting a bug in privileged software. Secure Objects can also be used to protect Virtual Machines in a system or Cloud Computing Environment that supports the concurrent execution of multiple Virtual Machines. This is useful in a Cloud environment to protect the confidentiality and integrity of one user's Virtual Machine from the applications and virtual machines of other users and from the Cloud Service Provider's software.

III. Solution Overview

As described above, a secure virtual machine (SVM) provides a cryptographically protected sandbox with confidentiality and integrity guarantees. We leverage SVMs to provide a safe remote environment for browsing links and attachments. When a user clicks on a link or an attachment, a Cloud-based service opens this link in a newly allocated SVM in the cloud instead of on the local system. Once the user completes browsing the link, the SVM is discarded. Figure 5 shows a high-level overview of the solution.

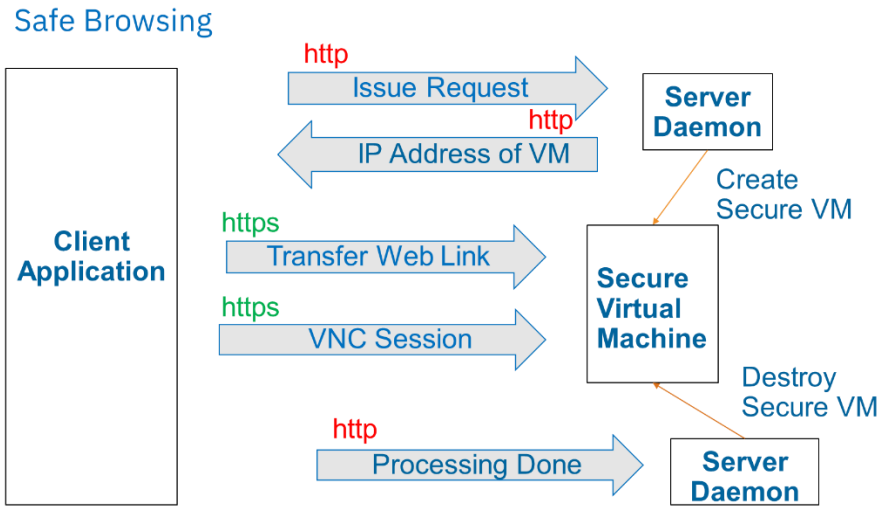

Figure 5. High-level description of safe web browsing protocol leveraging Secure VMs provided by a cloud service

The protocol provides the following security guarantees

- Each newly allocated SVM runs exactly the software that it's supposed to run. This allows one to know with a high level of confidence that an SVM that is provided to a user is unadulterated and free of malware when provided to the user. Since the Secure CPU technology protects the integrity of an SVM from other software, each newly-allocated SVM can be free of malware even if other software in the Cloud Service Provider's infrastructure -- including privileged software such as a hypervisor, operating system or device driver -- has been compromised.

- The Secure CPU technology and the Secure Virtual Machines allow one to know with a high level of confidence that an SVM that is launched to process an attachment or visit a web site does not contain spyware that "snoops" on the user or sends his sensitive or confidential information to a 3rd party.

- The Secure CPU technology and the Secure Virtual Machines also allow one to know with a high level of confidence that any public keys or digital certificates that an SVM may use to validate the identity of web sites have **not** been compromised. This is important since it allows a user to know with a high level of confidence that he really is connected to his bank's web site, say, and not to a fraudulent web site that has been set up to collect customer credentials and account information.

- The Secure CPU technology and the Secure Virtual Machines also allow one to know with a high level of confidence that any sensitive information that is processed within an SVM will not be accessible by any other software running in the Cloud even if privileged software such as a hypervisor or operating system, has been compromised.

- When an SVM is de-allocated, the Secure CPU technology makes it possible to determine whether the attachment that was processed or the web site that was visited left behind any malware -- and it enables this determination to be made *with a high degree of confidence*. Since the Secure CPU technology protects the integrity of a Secure Virtual Machine from other software:
    - one can know exactly what was in an SVM when it was launched to process an attachment or visit a web site, and
    - one can know, when an SVM is de-allocated, that any differences in the SVM from the norm were the result of the attachment that was processed or the web site that was visited. This information can then be passed on for a more in-depth analysis, which can be used to fix bugs or to blacklist web sites or sources of email, and it can also be passed on to appropriate law enforcement authorities.

Note that since a VM is discarded after the link/attachment has been processed, the malware remains confined to the VM *unless* it can undermine the application that is used for transferring results (in our case, the desktop sharing application) to the user. Fortunately, since this is a single application, this protocol reduces the attack surface for the malware significantly. A quantification of this reduction is considered future work.

IV. **Implementation**

A proof-of-concept implementation of the service described above contains the following three components:

1. A browser extension
2. A server-side application for creating SVMs.
3. A startup script running inside the SVM

Browser Extension: A browser extension on the client allows a user to indicate, via a right-click, that he or she would like to open a link or attachment in a secure VM. The extension performs the following steps:

1. The extension listens for a right-click to open a link or attachment in a secure virtual machine.
2. On a click, the extension calls an event handler which establishes a connection (over http) with the server-side application on the cloud to request a secure virtual machine.
3. Once the server-side application indicates that the secure VM has been allocated, the extension establishes a secure connection (over https) with an application running on the secure VM.
4. The extension transfers the web link to the Secure VM.
5. When the Secure VM accesses the link, the extension starts a browser based VNC session to the SVM over the encrypted connection. This allows the client to browse the link/attachment processed in the SVM through the VNC session.
6. Once the client finishes browsing the link, an event handler in the extension signals the server-side application to destroy the allocated secure VM.

Server-Side Application: The cloud service provides the server-side application and performs the following steps:

1. The application listens for connections from clients

2. On a client request, the application allocates a new secure VM from a master image that runs trusted software that is free of any known malware.
3. The application returns the IP address and port of the startup application running on the SVM.
4. Once a client indicates that an SVM is no longer required, the application discards the SVM.

SVM Startup Script: The SVM startup script is launched at boot-time when a new SVM is created from the master image. The script performs the following steps:

1. The script waits for a secure connection (over https) from a client.
2. Once the client transfers the web link, the script opens this link (either a web page or an attachment) in its local browser.
3. The script starts a VNC server that the client connects to.
4. Once the client closes the connection with the SVM, the SVM destroys the certificate that is used by the SVM to prove authenticity to the client over https (to be implemented in future versions). This ensures if a malicious server-side application reallocates the same SVM to a different client in the future, the SVM will not be able to prove its authenticity to the client, in which case the client will not proceed with the protocol.

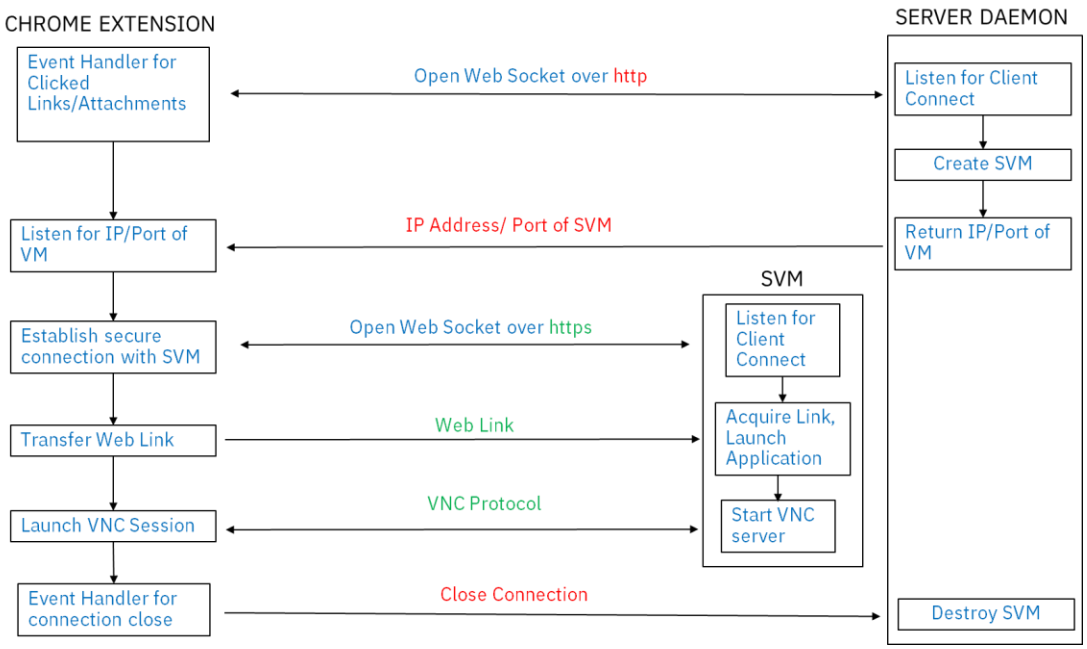

Figure 6. Description of the pipeline to provide a safe web browsing session for Google Chrome clients.

Figure 6 shows a detailed description of the browser extension (in this case for Google Chrome) and the server-side Daemon and the startup script in the SVM.

V. Evaluation

Test Bench – In our initial prototype, we implement and evaluate the safe browsing protocol on a local test bench since Secure VMs are still under development. Once the infrastructure for deploying Secure VMs in the IBM cloud is in place, the prototype can be ported to use Secure VMs instead of regular VMs as used in the current version.
The client side Google Chrome extension runs on Google Chrome version 56.0 and higher and is independent of the underlying OS. The server-side application has been implemented in C and runs on a Linux box with Intel i7 -3720QM CPU running at 2.60 GHz with 15.3GB RAM. The underlying OS is Ubuntu 14.04. The server-side application creates and destroys virtual machines through KVM, a standard virtualization solution for Linux on x86. The VMs run Ubuntu 14.04 and are allocated 4GB of RAM. The VMs are provisioned with Google Chrome 56.0+ and x11vnc for the VNC server.

Browsing Overhead – The browsing overhead is mainly due to the time required to create a VM, wait for it to boot and then establish the secure communication channel. In this case, the overall time required to open a link/attachment in a VM is around 45s. This overhead is significantly reduced when the server-side application can create and destroy VMs in parallel to processing requests as well keeping a pool of VMs ready to process client requests. In this case the time required to open a link in a secure VM is on an average 6s. This is mainly due to the overhead for establishing the secure socket connection. In a real-world cloud deployment where both the server and VMs can be overprovisioned, the protocol can be implemented with lower overheads and thus ensure that safe browsing can be supported with acceptable overheads.

VI. Future Work

Secure VMs for Sensitive Computation - This work shows how to leverage secure VMs for safe browsing of web links and attachments. As an extension, other forms of sensitive computation can be performed in a secure VM, subject to the condition that the secure VM contains an application to perform the computation. In fact, a secure VM provides a safer environment for processing sensitive information than a client device with limited protection, that may be infected with malware.
Since the SVM ensures both confidentiality and integrity of the environment, the client can offload sensitive computation to the cloud service with a high degree of confidence that the information will not be leaked to untrusted parties.

Analysis of Attack Surface – In this work, we significantly reduced the attack surface for malware to only the application used for transferring results from the secure VM to the client application (in our case through a remote desktop sharing session). We believe that this is a significant step forward over existing approaches where the attack surface is the entire client software stack including privileged software like the OS, Hypervisor etc.
As future work, it will be interesting to quantify and analyze this attack surface and to mitigate any existing vulnerabilities in this attack surface.

Client-Side Verification – To provide the above described service, it is desirable to also ensure that secure VMs can authenticate clients either through certificates or through a challenge-response protocol based on shared secrets. This will allow the model to handle untrusted clients.

Secure Containers – If only certain applications (or parts of applications) are outsourced by the client (for example, in our case the client outsources web browsing), then "secure containers" can provide a more lightweight solution. This will require analyzing the design of such secure containers to provide the required service. Alternatively, secure VMs can be created with minimal VM images to support only the functionality required for processing client requests.

VII. Related Work

Browsing web links/attachments safely and in general protecting devices from unintentional downloading of malware from the web has been well researched. Previous solutions can be broadly classified into four categories

Content-Aware Malware Detection - This category of work includes commercially available anti-virus/anti-malware software that try to detect the presence of malware on a device. Similar technology exists for browsing the web e.g. FireEye Email Security[7] applies sophisticated heuristics and malware-detection algorithms to uncover advanced malware in Web pages as well as across various file types. Sessions are replayed in a (safe) virtual execution to determine whether the suspicious traffic actually contains malware. Unfortunately, such techniques are not foolproof solutions since malware is often designed while incorporating sophisticated techniques to evade detection by commonly available software. Thus, this method of malware detection is vulnerable to zero-day attacks.

Blacklists - Blacklists are frequently used by browsers to track already known malicious websites. The browser blocks a user trying to access a website that is part of the blacklist to protect the user's device from potential malware being hosted by the website. Examples of such blacklist based solutions are Google Safe Browsing[8], McAfee Site Advisor[9] etc. Unfortunately, blacklists only include websites that are known to be malicious based on prior knowledge. Since, new malicious websites are created at an ever-increasing rate, it is non-trivial to keep track of all malicious websites. Consequently, blacklists are vulnerable to zero-day attacks

Whitelists – In contrast to blacklists, whitelists intend to track web sites that are known to be safe and allows browsing only to sites that are on the whitelist. Examples of whitelists include Bit9[10] and CoreTrace. Unfortunately, it is non-trivial to keep an updated whitelist. Further, a program/website currently on the whitelist can become infected and remain on the whitelist. In this case, all entities that trust the whitelist will be exposed to this vulnerability.

Content-Agnostic Malware Protection – Content-Agnostic malware protection such as CAMP[11] use reputation based systems to predict whether a website/software is malicious. The reputation of a given website/software is evaluated based on multiple known instances of visits/downloads. Further multiple features are used to determine whether a website is safe and overcome the challenges of maintaining an updated blacklist/whitelist. Unfortunately, reputation based systems can result in false negatives where an unsafe website is marked as safe. Similarly, false positives can flag a safe website as unsafe. Thus, the user may either visit an unsafe website because of the false negative or may be unable to visit a safe website as a result of the false positive, both adversely affecting the user's security and experience.

Citrix Safe Browsing[12] and FireGlass Web Isolation[13] are other examples of Content-Agnostic Malware Protection. Citrix provides a service similar to the one described here that allows the opening of unverified content in a separate, disposable virtualized environment using a temporary virtual machine (VM). However, they do not provide security guarantees for their virtual machines similar to those discussed in this paper. Thus, without confidentiality and integrity protection of the virtual machines, client data can be leaked to malware. Also, since the VMs are not secure, they cannot store secrets that can be used to prove their authenticity to clients without trusting the host. In other words, the host needs to be trusted to create VMs that run a trusted software stack. To remedy this problem, Citrix suggests "cleaning" the host machine of malware periodically. Unfortunately, this still exposes a significant window for malware to attack client devices. In contrast, in this work, we do not need to trust the host. The scheme remains secure even with an infected host since the secure VMs are secure even if the host is infected. Furthermore, the secure VMs ensure the confidentiality and integrity of client data. Fireglass provides a service similar to Citrix but, like Citrix, it does this without the strong security guarantees discussed in this report.

Bibliography:

1. https://www.symantec.com/connect/blogs/latest-intelligence-july-2017
2. https://www.forbes.com/sites/leemathews/2017/07/17/new-password-stealing-malware-spreads-rapidly-thanks-to-rock-bottom-pricing/#27eb41b36f16
3. Kaspersky Security Bulletin, 2016
4. Forbes Cyber Security, June 2017
5. The Guardian, July 2017
6. SecureBlue++: CPU Support for Secure Execution, IBM Research Report RC25287, May 23, 2012

http://domino.research.ibm.com/library/cyberdig.nsf/papers/E605BDC5439097F085257A13004D25CA/$File/rc25287.pdf
7. Fireeye Email Security https://www.fireeye.com/products/ex-email-security-products.html
8. Google Safe Browsing https://developers.google.com/safe-browsing/
9. McAfeee Site Advisor https://home.mcafee.com/root/landingpage.aspx?lpname=get-it-now&affid=0&cid=170789
10. Bit9 http://www.priveon.com/csamigration/
11. CAMP: Content-Agnostic Malware Protection; Rajab, Ballard, Lutz, Mavrommatis, Provos; NDSS '13
12. Separate, Disposable Execution Environment for Accessing Unverified Content, US Patent Application Publication 2015/0089497 A1, Borzycki et al., March 26, 2015
13. FireGlass, https://www.symantec.com/products/web-isolation