IEEE Consumer Electronics Magazine

MultiScan: a Scalable Online Virus Detection System with Multiple Anti-virus Engines

Journal:	*IEEE Consumer Electronics Magazine*
Manuscript ID	CEMAG-SRI-0003-Sep-2017
Manuscript Type:	Original Article
Date Submitted by the Author:	08-Sep-2017
Complete List of Authors:	Liu, Ming; University of Technology Sydney, School of Electrical and Data Engineering He, Yuxuan; Shanghai Jiao Tong University Xue, Zhi; Shanghai Jiao Tong University Chen, Jinjun; Swinburne University of Technology He, Xiangjian; University of Technology Sydney, School of Electrical and Data Engineering
Keywords:	Security < Network Technology, Privacy < Security and Rights Management, Trusted Computing < Security and Rights Management
Note: The following files were submitted by the author for peer review, but cannot be converted to PDF. You must view these files (e.g. movies) online.	
multiengineonlinevirus.tex mybibfile.bib	

MultiScan: a Scalable Online Virus Detection System with Multiple Anti-virus Engines

Ming Liu, Yuxuan He, Zhi Xue, Jinjun Chen, *Senior Member, IEEE*, and Xiangjian He, *Senior Member, IEEE*

Abstract—During recent years, computer viruses have evolved from traditionally simple forms to currently more sophisticated and stealthy "advanced persistent threats". Accordingly, the kind of virus detection systems that are composed by multiple anti-virus engines has arisen. This variety of systems such as VirusTotal and VirSCAN can offer the users with multiple detection results. However, those systems may save and distribute the user-uploaded samples to those security enterprises that own the anti-virus engines, which is not acceptable for the users with high privacy requirements. Considering this issue, in this article, we provide MultiScan, a comprehensive and scalable multi-engine online virus detection system, which incorporates 32 anti-virus engines and has a user-friendly web interface. The proposed system can perform the "offline detection and isolated update" of the anti-virus engines. This mechanism guarantees that the uploaded confidential samples are not exposed to the Internet, during either virus detection or system upgrade. Furthermore, the low-coupling design of this system is highly scalable to support the distributed deployment mode. The system testing results demonstrate that the mechanisms in MultiScan are efficient and practical.

Index Terms—Advanced persistent threat, virus detection, anti-virus engine.

1 Introduction

The battle between computer viruses and the anti-virus systems has last for decades. Nowadays, the intrusive cyber behaviors have become more complicated and diversified. Computer viruses have evolved from traditionally simple forms to currently more sophisticated and stealthy "advanced persistent threats (APT)", for the sake of hiding from the anti-virus systems. Fortunately, the anti-virus systems evolve simultaneously with the viruses. The anti-virus organizations currently may own a variety of advanced detection technologies and large-scale repositories of virus samples. However, for one particular anti-virus engine, false alarms or missed alarms during virus detection can still occur occasionally. Consequently, it has emerged as the concept of virus detection with a system that is composed by multiple anti-virus engines [1] [2] [3] [4]. This kind of system allows users to upload suspicious samples online and then presents the users with multiple independent detection results. Via the summarization and comparison of these results, both of the missed alarm rate and the false alarm rate can be reduced. The virus detection processes are usually conducted in virtualized environments with centralized control modules [5] [6] [7]. VirusTotal [8] and VirSCAN [9] are two signature systems that provide this kind of services. But these systems may save and share the user-uploaded suspicious samples to those security enterprises that own the anti-virus engines. This situation may cause some privacy-leakage problems, in particular for the samples with high confidentiality, which are not acceptable for the users with high privacy requirements. Therefore, it is necessary to construct a private and online virus detection system with multiple anti-virus engines that can provide accurate virus detection services for those users with high privacy requirements. Considering this issue, in this article, we provide MultiScan, a comprehensive and scalable multi-engine online virus detection system, which incorporates 32 leading anti-virus engines. With the user-friendly web interface, the system users can upload suspicious samples via web browsers and the detection results from multiple anti-virus engines can be displayed on the web interface. The proposed system can perform the "offline detection and isolated update" of the anti-virus engines. This mechanism guarantees that the uploaded confidential samples are not exposed to the Internet, during either virus detection or system upgrade. Furthermore, the low-coupling design of this system is highly scalable to support the distributed deployment mode. The system accomplishes the following aspects of functionalities:

1) *Combination of multiple anti-virus engines*. The system incorporates 32 anti-virus engines. As the anti-virus engines are closely related to the operating systems, multiple anti-virus engines installed in the same operating system may cause conflicts. Therefore, the anti-virus engines are installed in different virtual machines and controlled by customized scripts to ensure that each of them can work independently and efficiently. After the suspicious samples have been uploaded to the system, each engine will perform virus scan individually and report the detection result, hence finally 32 results will be returned to the user. The anti-virus engines can be automatically updated.
2) *User-friendly web interface*. As with VirusTotal, the system users can get access to the web interface and upload sus-

- *Ming Liu, Yuxuan He, and Zhi Xue are with the School of Cyber Security, Shanghai Jiao Tong University, China. E-mail: ming.liu-2@student.uts.edu.au*
- *Jinjun Chen is with the School of Software and Electrical Engineering, Swinburne University of Technology, Australia. E-mail: jchen@swin.edu.au*
- *Xiangjian He is with the School of Electrical and Data Engineering, University of Technology Sydney, Australia. E-mail: Xiangjian.He@uts.edu.au*

Manuscript received ??; revised ??.

picious samples (either separately or within a compressed archive) via web browsers and the detection results from multiple anti-virus engines can be displayed on the web interface. The system administrator can get access to the administrator's web interface for the related functionalities. The web interface also provides the API for the articulation with other third-party tools.

3) *Efficient user management*. The system has a critical access control mechanism. Ordinary users cannot get access to the administrator's web interface, with which the system administrator can monitor some factors such as the uploaded samples, the computational resources, the status of anti-virus engines, and the statistics of detection results. User information such as the username and password can be modified by both of the user and the system administrator, yet only the system administrator can add or delete users. As the system is designed only for the internal usage, registrations from the public are not allowed.

4) *Fault tolerance and security*. The interfaces for calling 32 anti-virus engines provided by different anti-virus enterprises are miscellaneous, and the update of these anti-virus engines may alter the interfaces as well. In this case, the system should handle all possible unstable factors properly and report them to the system administrator. The system includes multiple subsystems to control the anti-virus engines and their update collaboratively. The detailed logs are recorded by the system, and the error messages can be returned to the administrator timely for troubleshooting. Besides, the system has been scanned by a vulnerability scanner to avoid common vulnerabilities.

The rest of this article is composed as follows. In section 2, we describe the virus detection process of the proposed system. In section 3, we introduce the system design specifications. In section 4, we present the system testing process. In section 5, we provide the conclusion of this article.

2 The Virus Detection Process

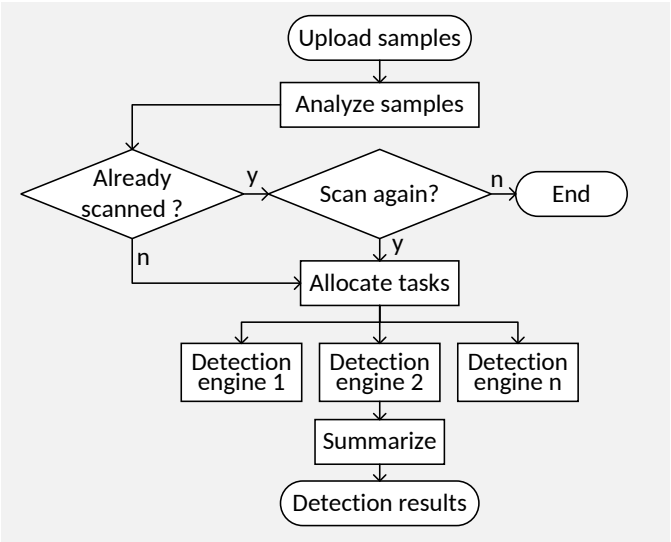


Fig. 1. The virus detection flow chart.

The system performs the virus detection process according to the flow chart illustrated in figure 1. Users can simply get access to the system via the web interface in the front-end. The web interface takes the responsibility of interacting with system users, such as accepting uploaded samples. After logging into the system with the username and password, a user can upload a single sample or multiple samples in a compressed archive via the web browser, either with a PC or a mobile device. E.g., a user can upload the newly downloaded APK files to the system for detection. The web interface provides JSON API, with which the third-party applications can upload samples to the system. After the samples are successfully uploaded, the system will perform immediate and automatic sample analysis. For an over-sized sample, the system can prompt an error message on the web interface, informing the permitted size of maximum. For an archive of samples, the system can perform the unzip process automatically and create a batch task; if the archive contains an excessive number of samples, the system can also prompt an error message on the web interface, informing the permitted number of maximum. If a sample has been scanned before, the system can inform the user about this situation and provide the detection history; the user can also request the system to scan the sample again. For the virus detection, the SHA-1, SHA-256 and MD5 hash values of the samples are generated and saved. Then the detection tasks are created based on the status of anti-virus engines. Each anti-virus engine is independently installed in a virtual machine and separated from other anti-virus engines, and it is packaged with a customized management script. When a detection task is allocated to one of the anti-virus engines, the script calls the engine to scan. Finally, multiple detection results are collected using the APIs of anti-virus engines, and the results are summarized and displayed on the web interface.

3 System Design Specifications

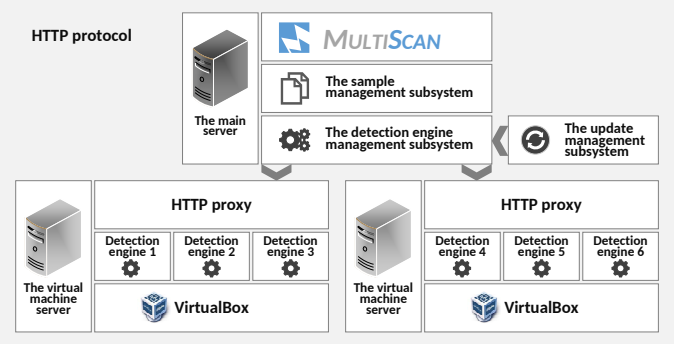


Fig. 2. The system architecture.

The system architecture is depicted in figure 2. The system has a user-friendly web interface in the front-end and the comprehensive anti-virus mechanisms based on virtualization techniques in the back-end. Ubuntu Linux is the primary operational environment of the proposed system and is used for the storage of suspicious samples, as the performance stability and security of Linux are better than Windows for the proposed system. Apache2 is utilized for building the web server, accompanied with Laravel PHP

TABLE 1
Web pages of the interface and their functionalities

Class	Web Page	Functionalities	User Types
The web pages for virus detection	Home page	Introduction of the system.	All users
	Upload page	Upload suspicious samples.	Logged-in users, system administrator
	Result page	View the detection results of one user's uploaded samples.	The logged-in user who uploaded the samples, system administrator
The web pages for system and user management	User center page	Modify the password, renew the access token, etc.	Logged-in users, system administrator
	Upload history page	View all uploaded samples of one user and the relevant detection results.	Logged-in users, system administrator
	Full history page	View all uploaded samples of all users and the relevant detection results.	System administrator
	User management page	Add or delete users, modify the passwords of all users, etc.	System administrator
	System status page	Check the status of CPU, RAM, and network, etc.	System administrator
	Statistics page	Check the statistics, such as the number of samples uploaded daily; the average detection time of all anti-virus engines, etc.	System administrator

and MySQL. The open-source software VirtualBox is utilized for virtualization. The system design is low-coupling, for the communications among the system modules are launched with HTTP protocol. Thus the high-scalability can be achieved. The system can execute not only on one physical host but can be distributed to multiple hosts as well. The troubleshooting can also be easily conducted with this design: if a system module encounters a fatal error, other modules will not be severely affected; after the error has been fixed, all modules can be started up again and execute normally. In overall, the system is composed of five core modules, including:

- *Web interface*. The web interface is designed for the interactions with the system users.
- *Sample management subsystem*. This subsystem manages the uploaded suspicious samples and supervises the allocation of virus detection tasks.
- *Engine management subsystem*. This subsystem controls and communicates with the anti-virus engines and collects their status of execution.
- *Anti-virus engines*. Each anti-virus engine is enclosed in one virtual machine. These anti-virus engines can communicate with the engine management subsystem via their customized scripts.
- *Update management subsystem*. This subsystem manages the "offline detection and isolated update" process. This subsystem controls the network connections of the virtual machines, and it controls the creation and deletion of the virtual machine snapshots.

3.1 The web interface

The front-end web interface is a unified API of the system for the interactions with the system administrator, system users, and third-party applications. After logging into the system with the username and password via the web interface, a user can upload a single suspicious sample or multiple samples compressed in a ZIP or RAR archive. After the samples are successfully uploaded, the system will perform the virus detection automatically. If a sample has been scanned before, the relevant detection history can be retrieved, and also the sample can be re-scanned. The detection results of multiple anti-virus engines can be summarized and displayed in the web interface for the users. On the management pages of the web interface, the system administrator can check the system executional status and conduct the user management. The design style of the web interface is concise and similar to VirusTotal. Figure 3 represents the web page for uploading suspicious samples.

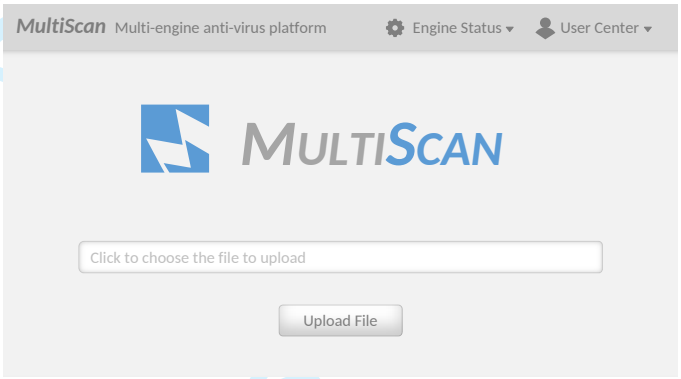


Fig. 3. The web page for uploading suspicious samples.

Compatibility is the main advantage of building the system interface on web browsers. The browser-server design ensures that the accesses of users are not limited by the hardware and operating systems. Users can also get access to the system using mobile devices. E.g., if a user has an Android smartphone, the newly downloaded APK files can be uploaded to the system immediately for virus detection. However, the web interface requires the PC browsers to be Chrome, Firefox, Safari, IE8 or above. For mobile devices, Android and iOS built-in browsers are applicable. As the web interface is the entry of the whole system and the control terminal for both of the regular users and system administrator, the relevant access control mechanism is essential. In the proposed system, the web interface is composed of multiple web pages, which have various functionalities. The accessible web pages for regular users and the system

administrator are different. Table 1 describes the web pages of the interface and their functionalities. A regular user can access limited functionalities, including:

- Check the detection history of the user-uploaded samples.
- Modify the account information and password.
- Generate the access token.
- Check the status of anti-virus engines.

Besides the above user functionalities, the system administrator can access some additional functionalities, including:

- View the uploaded samples of all users.
- Check the executional status of the whole system.
- View the statistics of samples, such as the number of samples uploaded daily.
- Conduct user management, including add or delete users, and modify their information or passwords.

JSON API is provided in the system for third-party applications to facilitate the process of uploading the samples and retrieving the detection results automatically. JSON API is based on HTTP, and the information is transmitted in JSON format. As the proposed system utilizes RESTful API, the third-party applications can upload samples, check the anti-virus engine status, and retrieve the detection results with standardized approaches. After logging-in, on the user center page, a user can generate the access token required for JSON API access. The code below represents the status of anti-virus engines in JSON format.

```
{
  "result": "success",
  "engines": [
  {
    "name": "f-prot",
    "full_name": "F-Prot",
    "platform": "Windows",
    "library_date": "01-05-2017",
    "status": "online"
  },
  ...
  {
    "name": "symantec",
    "full_name": "Symantec Endpoint Protection",
    "platform": "Windows",
    "library_date": "02-05-2017",
    "status": "online"
  }]
}
```

The "engines" part represents the status of anti-virus engines in the system, including the name, the operational status (online, offline, or updating), the operating system, and the update date of virus library. The code below represents the standardized detection results returned.

```
{
  "results": {
    "symantec": {
      "result": "OK",
      "full_name": "Symantec Endpoint Protection",
      "library_date": "02-05-2017"
  },
  ...
    "f-prot": {
      "result": "Virus",
      "full_name": "F-Prot",
      "library_date": "01-05-2017"
    }
  },
  "total": 32,
  "nothreatsNum": 20,
  "errorNum": 0,
  "detectedNum": 12
}
```

The "results" part represents the results of the anti-virus engines after a sample has been uploaded to the system. If a result is "OK", then the sample is benign for the relevant anti-virus engine; if a result is "TIMEOUT" or "ERROR", then the corresponding anti-virus engine encounters errors or malfunctions; if a result is "Virus", then the sample is malicious. The "nothreatsNum" represents the number of engines that report benign. The "errorNum" represents the number of engines that encounter errors or malfunctions. The "detectedNum" represents the number of engines that report malicious.

3.2 The sample management subsystem

The sample management subsystem manages the uploaded samples from users. After the samples are successfully uploaded, the sample management subsystem will perform sample analysis instantly. For an over-sized sample, the subsystem can prompt an error message on the web interface, informing the permitted size of maximum. For an archive of samples, the subsystem can perform the unzip process automatically and create a batch task; if the archive contains an excessive number of samples, the subsystem can also prompt an error message on the web interface, informing the permitted number of maximum. The SHA-1, SHA-256 and MD5 hash values of the newly-uploaded samples are automatically generated and saved into the Linux storage system, where the samples are searchable according to their hash values. Then the detection tasks are generated based on the status of anti-virus engines. Users can check the detection status of the samples online. If a sample has already been scanned, then the relevant detection history can be retrieved, or the sample can be re-scanned.

3.3 The engine management subsystem

As the system is a combination of 32 anti-virus engines from various anti-virus organizations, the most instable factor is the condition of these anti-virus engines. As these anti-virus engines are developed by various companies, the functionalities are different, and they require different operational environments. Uncertain detection results may be generated due to the instability of execution or update. Therefore, it is critical for the status monitoring and error handling of these anti-virus engines to make sure that they can execute and update efficiently and proper detection results can be returned. The engine management subsystem communicates with the scripts of the anti-virus engines with HTTP protocol and manages these engines. This subsystem can monitor and detect the errors of the anti-virus engines and inform the system administrator timely. Information such as the status of anti-virus engines and the versions of the virus libraries are collected by this subsystem.

3.4 The 32 anti-virus engines and packaging scripts

The system adopts the open-source software VirtualBox as the virtualization platform. Each of the 32 anti-virus engines is installed in an independent virtual machine and isolated with others so that they can execute independently and the conflicts can be avoided. With the low-coupling design, the system can still execute normally even if some of the anti-virus engines fail. The virtual machines can be distributed among multiple physical hosts. Operating systems such as

TABLE 2
Packaging methods of 32 anti-virus engines

Anti-virus Engines	Operating System	Engine Calling Method	Result Acquisition Method	Library Update Date Acquisition Method
Kingsoft, Rising	Windows	Command line	GUI software log	Configuration file
360	Windows	Command line	GUI software log	VDF file modification date
AVG, DrWeb	Windows	Command line	Detection report	Detection report
McAfee	Windows	Command line	Detection report	Configuration file
Norman	Windows	Command line	Detection report	VDF file modification date
BitDefender, eScan	Linux	Command line	Command line output	Command line output
Avast, ClamAV, Emsisoft, NOD32, F-Prot, F-Secure, IKARUS, Kaspersky	Windows	Command line	Command line output	Command line output
G-DATA	Windows	Command line	Command line output	Detection report
Comodo	Linux	Command line	Command line output	VDF file modification date
Microsoft	Windows	Command line	System event log	VDF file modification date
ZoneAlarm	Windows	Command line	Software log	VDF file modification date
Agnitum, Defenx	Windows	Trigger real-time protection	SQLite file	VDF file modification date
Avira	Windows	Trigger real-time protection	System event log	VDF file modification date
Symantec	Windows	Trigger real-time protection	System event log	Configuration file
K7, TrendMicro	Windows	Trigger real-time protection	Software log	VDF file modification date
Fortinet, Panda, Tencent	Windows	ShellMenu	GUI software log	VDF file modification date
Baidu	Windows	ShellMenu	GUI detection report	Configuration file
MalwareSecure	Windows	ShellMenu	SQLite file	VDF file modification date

Windows XP 32bit, Windows Thin PC 32bit, CentOS 6.6 64bit are installed into the virtual machines, according to the anti-virus engines' requirements of operating systems. The anti-virus engines are customized according to their features, and enclosed in packages with executable scripts, which have the following features and functionalities:

- The scripts are saved in the main server for centralized management and modification.
- The scripts are composed with Python and communicate with the engine management subsystem with HTTP protocol.
- The scripts can monitor the anti-virus engines in real-time, and report their status including any errors occurred to the engine management subsystem.
- The scripts request virus detection jobs from the engine management subsystem regularly. When a detection task is allocated to one of the anti-virus engines, the script of that anti-virus engine downloads the sample to the local virtual machine and calls the anti-virus engine to scan. The detection results are collected using the APIs of the anti-virus engines.
- The scripts can update the virus libraries regularly and report the virus library versions to the engine management subsystem.

The methods for realizing the functionalities mentioned above are various for different anti-virus engines. For example, when calling the anti-virus engines and collecting the results, some of the anti-virus engines use command line interfaces whereas others use windowed ones. Therefore, various kinds of scripts for packaging the anti-virus engines have been developed according to their features. Table 2 illustrates the packaging methods of 32 anti-virus engines.

3.4.1 Engine calling methods

- *Command line*. In Windows and Linux, some engines support command line interfaces for calling.
- *ShellMenu*. In the Windows system resource manager, each file has its shell context menu, and an anti-virus engine often adds in its menu item. Windows provide calling functions to a file's shell context menu. Therefore, for some engines which do not support command line interfaces, the ShellMenu method is utilized for calling the engines for virus detection.
- *Trigger real-time protection*. Most of the engines have real-time protection mechanisms. Whenever a file is copied into a hard-disk, it will trigger the real-time protection. Therefore, for those engines that do not support the command line or ShellMenu methods, the real-time protection triggering method is utilized.

3.4.2 Result acquisition methods

- *Command line output*. Most engines can return the standardized detection results in the command lines with the command line calling method.
- *System event log*. For some engines, the detection results are recorded into the system event logs. This method is often applied on those engines that utilize the calling method of real-time protection triggering.
- *Software log*. Some engines save the detection results into software logs or user application data logs.
- *SQLite file*. Some engines save the detection results into SQLite databases.
- *Detection report*. Some engines generate the detection results as official detection reports.
- *GUI software log*. For some engines, instead of outputting text-based detection results directly, graphical windows are displayed for the results. For those engines, the scripts will try to acquire the results by simulating the manual manipulating processes, such as software-clicking, keyboard-typing, screen-shotting, and image matching. The acquired results are text-based software logs.

3.4.3 Virus library update methods

- *Automatic update*. Most engines can check the network connections and communicate with the official update

servers. When there are available update packages, the virus libraries will be automatically updated.

- *Command line calling*. Some engines have independent functions for calling to start the update processes.
- *Offline update package installation*. Due to the network conditions, the online update of some engines may fail. In this case, the relevant offline update packages need to be downloaded from the official websites for the update.

3.4.4 Virus library update date acquisition methods

- *Command line output*. Some engines have command line interfaces, from which the dates can be acquired.
- *Configuration file*. Some engines record the dates into the configuration files.
- *Detection report*. Some engines record the dates into the official detection reports.
- *VDF file modification date*. For those engines whose update dates cannot be acquired from the above methods, the modification dates of the VDF files are regarded as the relevant virus library update dates.

3.5 The update management subsystem

Due to the privacy requirements of the system users, the samples should not be uploaded to the anti-virus enterprises. The "offline detection and isolated update" is an update technique that can perform the virus library update with the Internet connection, but the samples are isolated with the anti-virus enterprises. When scanning samples, the anti-virus engines are isolated from the Internet; and when updating, the anti-virus engines are isolated from the samples. With this technique, the anti-virus engines can be updated on a daily basis and there is no privacy leakage for the samples. This technique is monitored by the update management subsystem, which controls the operational status of the virtual machines, such as:

- The boot and shut down of the virtual machines.
- The network connection of the virtual machines (Internet or host only).
- The creation/deletion of the virtual machine snapshots.

This subsystem takes the snapshots named "scanner" from the virtual machines in which the anti-virus engines are installed and then sets the network configurations of the scanners to "host only" to block the Internet connections. For scanners, the packaging scripts will request virus detection jobs from the engine management subsystem regularly. During detection, all of the uploaded samples are only scanned by the anti-virus engines in these scanners. The original virtual machines are named "updater". The updaters have the Internet connections, but they are isolated to the uploaded samples. For updaters, the packaging scripts will launch the engine update processes on a daily basis to update the relevant virus libraries, and report the virus library versions to the engine management subsystem. After the virus libraries of the anti-virus engines in the updaters are updated, new scanners are generated from these updaters. The old scanners are then replaced by the new ones. This mechanism ensures that those anti-virus engines that connect to the Internet are isolated to the uploaded samples, and those anti-virus engines with the uploaded samples cannot get access to the Internet. Checkpoints are set during these operations, thus if some errors occur, they can be fixed, and the related error messages can be sent to the system administrator timely. Figure 4 demonstrates the complete process of the "offline detection and isolated update".

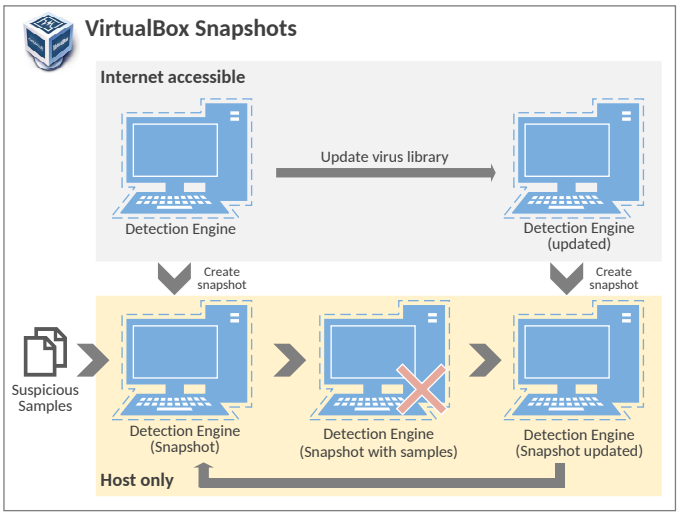


Fig. 4. The complete process of the "offline detection and isolated update".

4 System Testing Approaches

The objective of system testing is to guarantee that the system can work according to the requirements analysis. The system should present acceptable performance of fault-tolerance and privacy-preservation, and it should be defensive when confronted with regular cyber attacks. Via long-term testing, the stability of the anti-virus engines can be enhanced. The hardware requirement of the system can also be obtained via testing; thus the recommended hardware configuration can be correspondingly provided for the deployment in an enterprise. We have tested the system under the following hardware environment:

- Server: *Lenovo ThinkServer TD340*
- CPU: *Xeon E5-2420 v2 2.2GHz 4 cores*2*
- RAM: *32GB DDR3 1600MHz*
- Hard-drive: *Samsung 256GB SSD*2; 1TB 7200rpm HDD*2*

We use unit testing and end-to-end testing to guarantee the comprehensiveness of the testing process and thus the best performance of the system.

- *Unit testing* is the testing process of the subsystems, aims to ensure that each of the subsystems can execute normally and has no errors. Each of the five modules of the system has been tested.
- *End-to-end testing* simulates the behaviors of a system user, such as uploading samples and checking the detection results, aims to ensure that the system can execute all functions and processes normally in regular conditions.

For testing of the Web interface, we have simulated the user activities such as logging in, uploading samples, viewing the detection process, visiting the user center, and retrieving the detection history. The results have shown that the web interface layout is correct and the website can normally be accessed; all other functions can execute

normally. Figure 5 represents the screenshot of part of the detection result page.

Detection complete

MD5 Hash:	69630e4574ec6798239b091cda43dca0
SHA1 Hash:	cf8bd9dfdddf007f75adf4c2be48005cea317c62
SHA256 Hash:	131f95c51cc819465fa1797f6ccacf9d494aaaff46fa3eac73ae63ffBdfd8267
Detection result:	24/30

Detection engine	Detection result	Library date
K7	found the Trojan (000139291)	30-04-2017
Defenx	EICAR_test_file	30-04-2017
IKARUS	EICAR-ANTIVIRUS-TESTFILE	30-04-2017
ZoneAlarm	EICAR-Test-File	03-05-2017
Emsisoft	EICAR-Test-File (not a virus) (B)	30-04-2017
Agnitum Outpost	EICAR_test_file	29-04-2017

Fig. 5. The detection result of an uploaded sample.

For testing of the sample management subsystem, we have uploaded repeated samples, over-sized samples, and archived samples to the system. The results have shown that for a repeated sample, the system can inform the user and provide the detection history; if the user requests to scan the sample again, then the system will perform another detection process. For an over-sized sample, the system can prompt an error message with the permitted largest size. For an archived sample, the system can perform the unzip process automatically and create a batch task; if an archive contains an excessive number of samples, the system can also prompt an error message.

Status of Engines

Engine Name	Operating System	Date of Virus Library	Status
360 Antivirus	Windows	20/04/2017	Online
Avira Server Security	Windows	26/04/2017	Online
Malware Secure	Windows	26/04/2017	Online
TrendMicro PC-cilin	Windows	24/04/2017	Online
Rising Antivirus	Windows	25/04/2017	Online
Microsoft Security Essentials	Windows	27/04/2017	Online
F-Prot	Windows	24/04/2017	Online
Baidu	Windows	22/04/2017	Online
Tencent	Windows	21/04/2017	Online
Fortinet	Windows	22/04/2017	Online
Norman Antivirus	Windows	22/04/2017	Online
Panda Antivirus	Windows	21/04/2017	Online
Avast	Windows	22/04/2017	Online

Fig. 6. The status of anti-virus engines.

For testing of the engine management subsystem, we have designed multiple testing cases based on the anti-virus engines' status (online, updating, or offline). The results have shown that the subsystem can correctly obtain the anti-virus engines' status; via the collaboration with the sample management subsystem, the engine management subsystem can accurately assign detection tasks to the anti-virus engines, according to the anti-virus engines' status. This subsystem can record the error messages about the anti-virus engines and report them to the system administrator. Figure 6 represents the status of anti-virs engines collected by this subsystem.

For testing of the update management subsystem, a set of operations have been applied to the virtual machines, such as the generation and deletion of the virtual machine snapshots. The results have shown that the virtual machines can be started up sequentially; the snapshots can be generated and deleted properly. If the system is overloaded, no new snapshots will be created, and the related information can be sent to the system administrator.

For testing of the anti-virus engines and their packaging scripts, we have uploaded the malicious and benign samples and obtained the detection results. Benign files (such as Windows system files), EICAR Standard Anti-Virus Test File, and other suspicious samples collected from the Internet are used to test the accuracy and performance of the anti-virus engines. The results have shown that the anti-virus engines can normally work in regular conditions. The packaging scripts can detect errors of the anti-virus engines and return the error messages timely to the engine management subsystem for troubleshooting. The average overall detection time for a single uploaded file is 60 seconds, and 70 percent of the anti-virus engines can return the results in 30 seconds. For a batch task, the average overall detection time is 24 seconds per sample. Figure 7 demonstrates the average detection time of the 32 engines.

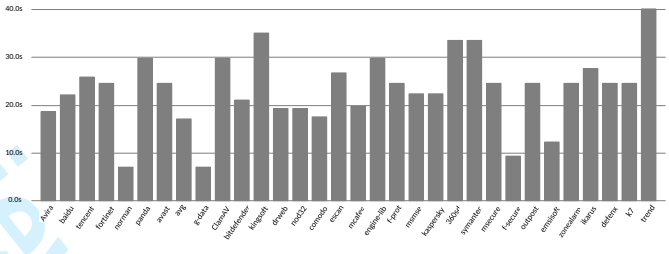


Fig. 7. The average detection time of the 32 engines.

Besides, we have specially tested the performance and security of the system. For the performance of the web interface, we used the Apache Benchmark to launch the tests. The results have shown that the website is available for hundreds of user accesses simultaneously. For the performance of the anti-virus engines and virtual machines, firstly all anti-virus engines are started up, then some archives (zip or rar) with more than one hundred samples are uploaded to the system for testing. The results have shown that for an archive with 119 samples, it takes 48 minutes to complete the task. The average detection speed is 24 seconds per sample. Although the load average of the system can exceed the number of CPU cores, the anti-virus engines can still work normally. According to the test results, for the overall performance of the system, the hardware environment in our testing only fulfills the lowest requirement. Thus, the recommended hardware configuration for the system is that each virtual machine should be allocated at least one CPU core and 1.5 GB RAM. Adequate storage space is recommended for the backup of virtual machine files. For the system security, we used the Acunetix Web Vulnerability Scanner to scan the system to guarantee that the system has no high-risk vulnerabilities. Besides, we audited all of the codes to avoid SQL injection and XSS vulnerabilities. We have simulated the unauthorized accesses of attackers. We also have simulated XSS and SQL injection attacks to the

system and tried to obtain the uploaded samples from the Internet. The results have shown that the system is defensive to these attacks.

Based on the above descriptions about testing, for the overall detection speed, the main limitation is the speed of each engine. As various companies have different engine design, the detection time varies enormously. It can be as quick as a few seconds and as slow as tens of seconds. Besides, some anti-virus engines use GUI interaction methods, which lower the speed. Therefore, according to the testing results, the system performance can be enhanced in three aspects, i.e., the improvement of web interface; the upgrade of system hardware; and the selection of more stable enterprise-edition anti-virus engines.

5 Conclusion and Future Work

We have proposed a multi-engine online virus detection system that can perform the "offline detection and isolated update". This mechanism guarantees that the uploaded confidential samples are not exposed to the Internet. With the user-friendly web interface, the system users can gain a comprehensive security evaluation of the suspicious samples based on the detection results from multiple anti-virus engines. The web interface also supports JSON API, thus the system can be integrated with other systems such as the intrusion detection system (IDS) or other security analytics software. Furthermore, the system is highly scalable to support the distributed deployment mode. The virtualization techniques used in the system facilitate the installation and execution of various types of anti-virus engines. The virtual machines can be scaled and installed to multiple physical hosts. As the system can collect and store the suspicious samples, in the future, we will further improve the system functionalities so that the system can analyze the distribution patterns of different kinds of malicious samples. Moreover, as the low-coupling design of the system shows ideal scalability, we will modify and build the system among several virtual machines in an IaaS cloud with cluster-based big data tools to further improve the system performance.

Acknowledgment

The authors would like to thank CSC for the support.

References

[1] A. Gilder, R. Herbst, and S. Shah, "Scan engine manager with updates," Nov. 6 2009, uS Patent App. 12/613,569.
[2] K. Shyamsunder, T. Tonn, R. Thomas, A. Holmes, J. Krahulec, and S. Sunkara, "Systems and methods for malware detection and scanning," Dec. 30 2010, uS Patent App. 12/982,540.
[3] M. D. McDougal, W. E. Sterns, and R. S. Jennings, "System and method for malware detection using multiple techniques," Apr. 14 2015, uS Patent 9,009,820.
[4] P. N. Yarykin and I. B. Godunov, "System and methods of performing antivirus checking in a virtual environment using different antivirus checking techniques," Jan. 19 2016, uS Patent 9,239,921.
[5] ——, "System and methods of distributing antivirus checking tasks among virtual machines in a virtual network," Mar. 24 2015, uS Patent 8,990,946.
[6] R. Thomas, M. LaPilla, T. Tonn, G. Sinclair, B. Hartstein, and M. Cote, "Systems and methods for malware detection and scanning," May 17 2016, uS Patent 9,344,446.
[7] J. Liu, X. Ouyang, and Q. Bo, "Dynamic malware analysis of a url using a browser executed in an instrumented virtual machine environment," Aug. 9 2016, uS Patent 9,413,774.
[8] VirusTotal, "Virustotal-free online virus, malware and url scanner," *Online: https://www.virustotal.com/*, 2017.
[9] VirSCAN, "Virscan-on-line scan service," *Online: http://www.virscan.org/*, 2017.

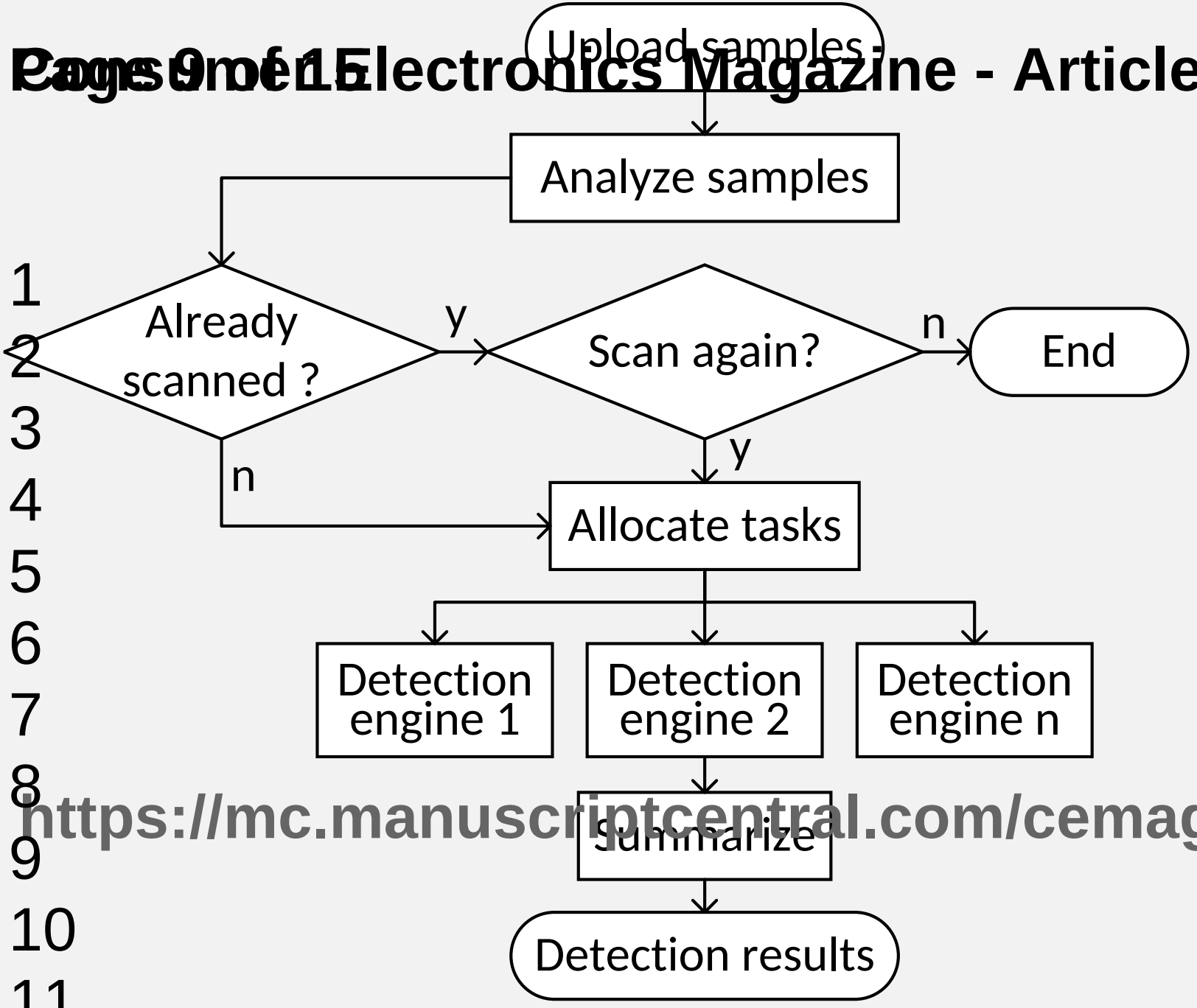
Ming Liu (ming.liu-2@student.uts.edu.au) is a joint PhD student at the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, China, and the Faculty of Engineering and IT, University of Technology Sydney, Australia. His research interests include cyber threat intelligence, intrusion detection systems, data analytics, and cloud security.

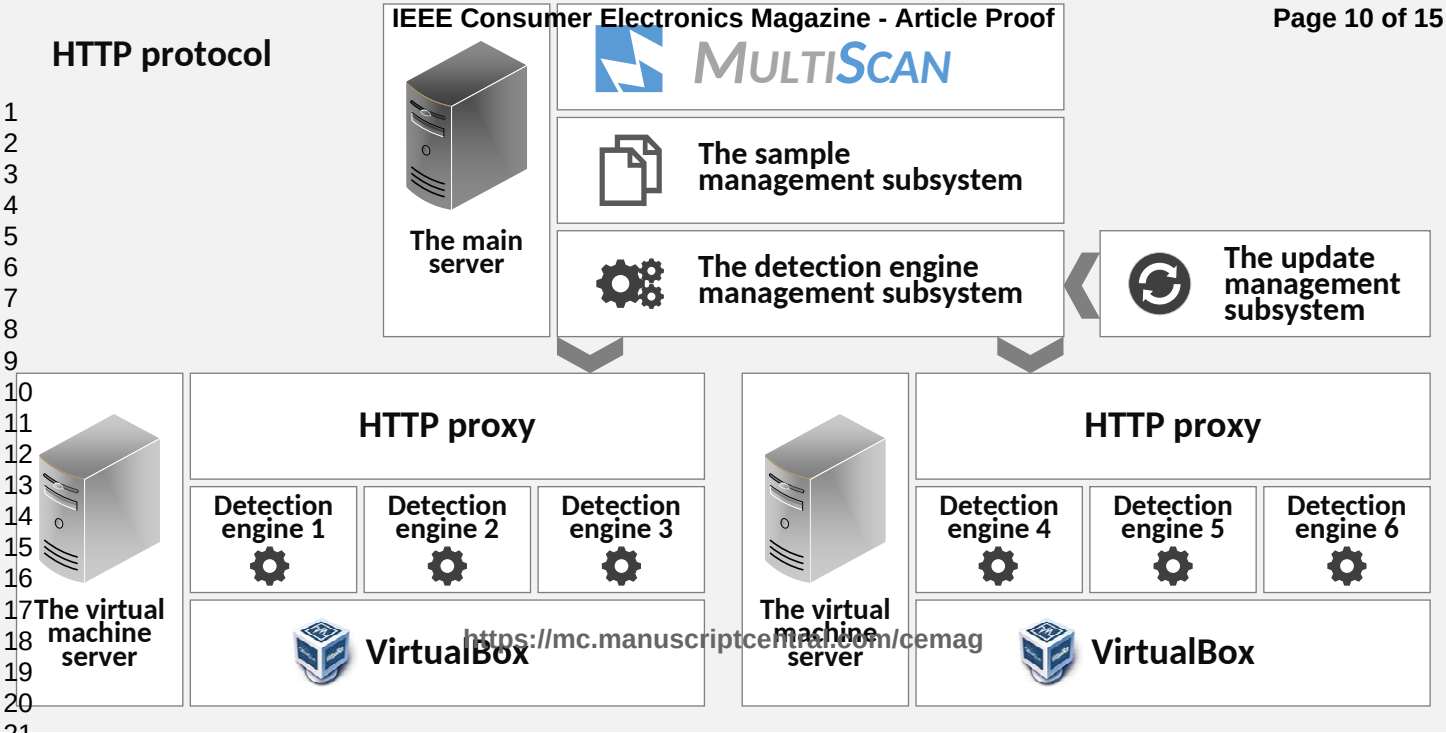
Yuxuan He (max.yuxuan.he@gmail.com) received his master's degree from Shanghai Jiao Tong University in 2015. Currently, he is a lead software engineer at Hypereal. His research interests include iOS development, cloud computing, GPU parallel computing, and network defense technologies.

Zhi Xue (zxue@sjtu.edu.cn) is a professor at the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, China, and the associate dean of the School of Cyber Security in the same university. He received his bachelor's degree from Shanghai Jiao Tong University in 1992, and was a visiting scholar in the Bell Laboratories of United States in 1997, and received his PhD degree from Shanghai Jiao Tong University in 2001. His research interests include wireless network security, cloud security, cryptography, and cyber threat intelligence.

Jinjun Chen (jchen@swin.edu.au) is a professor at the School of Software and Electrical Engineering, Swinburne University of Technology, Australia. He received his PhD degree from the same university in 2007. Currently, he is the deputy director of the Swinburne Data Science Research Institute. His research interests include data science, scalable data analytics, privacy preservation technologies, and cyber security. His research outcomes have been published in more than 100 articles in high-quality journals and conferences. He received the Swinburne Vice Chancellor's Research Award for early career researchers (2008), the IEEE Computer Society Outstanding Leadership Award (2008-2009), the IEEE Computer Society Service Award (2007), and the Swinburne Faculty of ICT Research Thesis Excellence Award (2007).

Xiangjian He (Xiangjian.He@uts.edu.au) received the Bachelor of Science degree in Mathematics from Xiamen University in 1982, the Master of Science degree in Applied Mathematics from Fuzhou University in 1986, the Master of Science degree in Information Technology from the Flinders University of South Australia in 1995, and the PhD degree in Computing Sciences from the University of Technology Sydney, Australia in 1999. Currently, he is a full professor and the Director of Computer Vision and Pattern Recognition Laboratory at the Global Big Data Technologies Centre (GBDTC) and a co-leader of the Network Security research team at the Centre for Real-time Information Networks (CRIN) at the University of Technology Sydney. He has many high-quality publications and has received various research grants including four national Research Grants awarded by Australian Research Council (ARC) as a Chief Investigator. He is an IEEE Senior Member and an IEEE Signal Processing Society Student Committee member. He has served as a guest editor for various international journals.





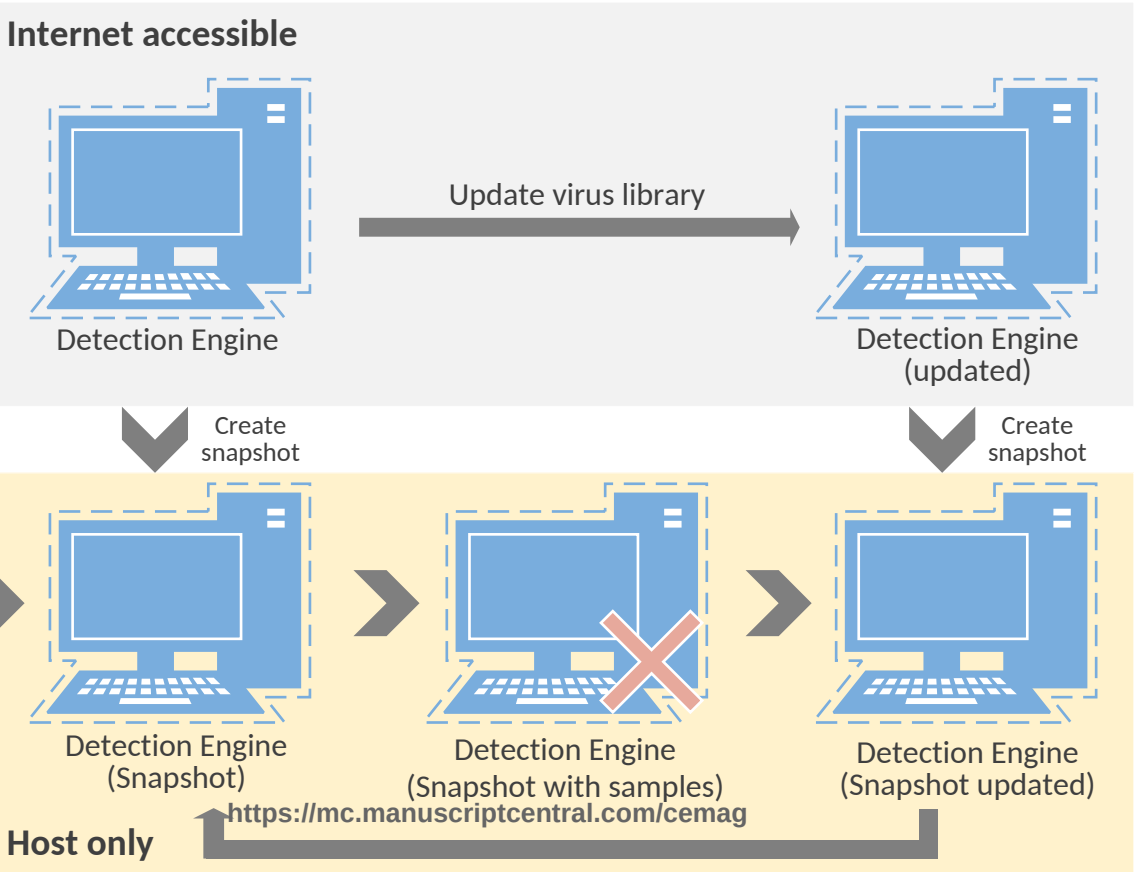
MULTISCAN

Click to choose the file to upload

https://mc.manuscriptcentral.com/cemag

Upload File

VirtualBox Snapshots



Detection complete

MD5 Hash:	69630e4574ec6798239b091cda43dca0
SHA1 Hash:	cf8bd9dfddff007f75adf4c2be48005cea317c62
SHA256 Hash:	131f95c51cc819465fa1797f6ccacf9d494aaaff46fa3eac73ae63ffBdfd8267
Detection result:	24/30

Detection engine	Detection result	Library date
K7	found the Trojan (000139291)	30-04-2017
Defenx	EICAR_test_file	30-04-2017
KARUS	EICAR-ANTIVIRUS-TESTFILE	30-04-2017
ZoneAlarm	EICAR-Test-File	03-05-2017
Emsisoft	EICAR-Test-File (not a virus) (B)	30-04-2017
Agnitum Outpost	EICAR_test_file	29-04-2017

Status of Engines

Engine Name	Operating System	Date of Virus Library	Status
360 Antivirus	Windows	20/04/2017	Online
Avira Server Security	Windows	26/04/2017	Online
Malware Secure	Windows	26/04/2017	Online
TrendMicro PC-cilin	Windows	24/04/2017	Online
Rising Antivirus	Windows	25/04/2017	Online
Microsoft Security Essentials	Windows	27/04/2017	Online
F-Prot	Windows	24/04/2017	Online
Baidu	Windows	22/04/2017	Online
Tencent	Windows	21/04/2017	Online
Fortinet	Windows	22/04/2017	Online
Norman Antivirus	Windows	22/04/2017	Online
Panda Antivirus	Windows	21/04/2017	Online
Avast	Windows	22/04/2017	Online