Flogger: A File-centric Logger for Monitoring File Access and Transfers within Cloud Computing Environments

K L Ryan Ko, Peter Jagadpramana, Bu Sung Lee

HP Laboratories
HPL-2011-119

Keyword(s):

Cloud computing; logging; auditability; accountability; trust in Cloud computing; trusted Cloud; Cloud computing security; file-centric logs; file-centric logging mechanisms; detective mechanisms

Abstract:

Trust is one of the main obstacles to widespread Cloud adoption. In order to increase trust in Cloud computing, we need to increase transparency and accountability of data in the Cloud for both enterprises and end-users. However, current system tools are unable to log file accesses and transfers effectively within a Cloud environment. In this paper, we present Flogger, a novel file-centric logger suitable for both private and public Cloud environments. Flogger records file-centric access and transfer information from within the kernel spaces of both virtual machines (VMs) and physical machines (PMs) in the Cloud, thus giving full transparency of the entire data landscape in the Cloud. With Flogger, services can be built above it to provide Cloud providers, end-users and regulators with the relevant provenance, e.g. a tool for an end-user to track whether his/her file was 'touched' by an unauthorized user. We present the initial developments of Flogger, and interesting results from our experiments. We also present compelling future work that will shape the beginnings of a new logging paradigm: distributed VM/PM file-centric logging.

External Posting Date: August 6, 2011 [Fulltext] Approved for External Publication
Internal Posting Date: August 6, 2011 [Fulltext]

Flogger: A File-centric Logger for Monitoring File Access and Transfers within Cloud Computing Environments

Ryan K L Ko, Peter Jagadpramana, Bu Sung Lee
Cloud and Security Lab
HP Laboratories Singapore
{ryan.ko | peter.jagadpramana | francis.lee}@hp.com

***Abstract*— Trust is one of the main obstacles to widespread Cloud adoption. In order to increase trust in Cloud computing, we need to increase transparency and accountability of data in the Cloud for both enterprises and end-users. However, current system tools are unable to log file accesses and transfers effectively within a Cloud environment. In this paper, we present Flogger, a novel file-centric logger suitable for both private and public Cloud environments. Flogger records file-centric access and transfer information from within the kernel spaces of both virtual machines (VMs) and physical machines (PMs) in the Cloud, thus giving full transparency of the entire data landscape in the Cloud. With Flogger, services can be built above it to provide Cloud providers, end-users and regulators with the relevant provenance, e.g. a tool for an end-user to track whether his/ her file was 'touched' by an unauthorized user. We present the initial developments of Flogger, and interesting results from our experiments. We also present compelling future work that will shape the beginnings of a new logging paradigm: distributed VM/ PM file-centric logging.**

Keywords- Cloud computing; logging; auditability; accountability; trust in Cloud computing; trusted Cloud; Cloud computing security; file-centric logs; file-centric logging mechanisms; detective mechanisms.

I. INTRODUCTION

Trust is one of the main obstacles to widespread Cloud Computing adoption. In order to increase trust in Cloud Computing, there are both preventive and detective measures [1]. While many Cloud Computing service providers are focusing on preventive measures (e.g. better firewalls, stronger encryption, etc), few are focusing on increasing the accountability and transparency of their Clouds via detective mechanisms (e.g. logging, reports for end-user self forensics) [2].

With Cloud computing removing the need for end-users to own systems, we also experience a change in mindset, from a focus on systems security to a focus on data security and protection. There is a need to know the "*who, what, where, when, how and why*" of data movements in the Cloud. This is made even more urgent with the impending data explosion [3], and the dawn of the so-called 'fourth paradigm' [3, 4] described by the late Microsoft researcher Jim Gray.

With the need for detective measures and the change in focus to data security and protection, comes a demand for a robust security tools which will enable end-users, Cloud computing service providers, administrators of Cloud services, and even regulators to inspect, monitor and analyze the trends of data accesses and movements within the large-scale Cloud computing environment from a single point of view. However, are current detective mechanisms ready for this change in paradigm? We begin by analyzing the current state of the art:

II. RELATED WORK

A. User Space Centralized File System Call Monitor

In traditional one-system or local area network (LAN) environments, it is common to find user-space file monitoring tools or extensions of file systems (e.g. iNotify [5], swatch [6], file alteration monitors (FAM) [7]) to be widely used for monitoring the single- or multiple-file activities within a single machine. Tools are also available for monitoring packets in networks (e.g. snort [8]). With large scales and heavy usage of virtualization technologies in Cloud computing, such tools are insufficient to provide an over-arching view for monitoring files across both virtual machines (VMs) and physical machines (PMs). Moreover, these applications are usually housed within the user space, leaving them vulnerable to user space attacks.

B. File Integrity Checkers as Intrusion Detection

File integrity checkers such as TripWire inspect for changes to the files in the systems by checking against a baseline hash-key database which is regularly updated with the latest hash keys of the files within a system. Such an implementation is not scalable for the Cloud as there is a high volume of access, i.e. the need to regularly update the key database is not feasible. Furthermore, these tools do not provide a history of the file changes. Hence, while they are able to identify which files have changed, they are unable to explain the history of what actually happened to the files. Such limitation is not desirable for forensics in the context of the Cloud.

C. Virtual Environment Monitors

With the rise in adoption of virtualization technologies especially in private Clouds, software such as the HyTrust Appliance [9] are starting to become more prominent. These tools enable administrators to regulate the access rights and to have an overview of the activities and consolidation of common system logs for all virtual machines. However, this visibility of the virtual layer is still not the full transparency requested by end-users [10] surveyed by the Fujitsu Research Institute, which states that 88% of these users want to know ‘exactly what goes on’ in the physical servers hosting the guest machines.

D. Cloud Systems Health and Performance Monitoring

When there is mention of monitoring, there is a current emphasis of monitoring the server performance in Clouds. Such a focus on system monitoring is not totally aligned to the actual needs of users. Despite having color schemes, visualizations and attractive dashboards, tools such as VMWare vFabric Hyperic [11] and CloudKick [12] are still unable to offer the crucial need of monitoring data movements and transfers in the Cloud.

III. NEW BREED OF LOGGERS REQUIRED

It is now evident from observing the limitations of the state-of-the-art that we need the following *necessary requirements* for effective monitoring of data in the Cloud:

- *Transcend VM/ PM* - It must be in kernel space, and must be able to transcend both virtual and physical spaces in the Cloud, providing full transparency of all operations in the Cloud.
- *Provenance* - It must provide a full or a summarized/ concise provenance of data life cycles and transfers in the Cloud. This is also in tandem with the increase in the emphasis of data governance [13] and accountability [1].
- *Single Auditable View* - It must be able to provide a single consolidated report for inspection.
- *Efficient storage* - It must be efficient in both short term storage and long term archival.
- *Analytics* – It must provide auditing features to enable strong analytics and quick observations of footprints of file activities and transfers.

With the above list in mind, we propose **Flogger** (short for File-Centric Logger), a novel file-centric logger that can be implemented in both VM and PM kernels in a non-invasive manner within nodes in the Cloud.

IV. FLOGGER - ARCHITECTURE AND DESIGN

A. Flogger Addresses the System Layer of the TrustCloud Framework

Flogger addresses the needs of system layer within the TrustCloud Framework [1]. TrustCloud is a layered framework describing the different layers of granularity for Cloud accountability. The System Layer in the framework highlighted the importance of monitoring and auditing containers of data (e.g. files) within and out of the Cloud.

With the foundational System Layer, we can then study movement and changes of data within and across files (Data Layer), and also workflows and data flows (Workflow Layer)– thus giving full provenance of data in the Cloud and in compliance to the Law/ Regulation Layer and the Policies Layer. Further descriptions of issues related to these layers of accountability are described in [1].

B. Flogger Components and Architecture

Figure 1 shows Floggers and their accompanying components, and demonstrates the underlying mechanisms capturing file actions and movements from the underlying kernel space (depicted by the numeric sequence in Figure 1). A simple example of the resulting file-centric log (in short, “flog”) captured by both a VM and its host PM is shown in Figure 2.

1) Components

The typical implementation consists of the following components (See Figure 1):

- Flogger (Linux) – A Linux Loadable Kernel Module (LKM) running on VM which intercepts file and network operations and writes the events as VM flogs.
- Flogger (Windows) – A Windows Device Driver running on PM which intercepts file operations and writes the events as PM flogs.
- Components accompanying Flogger
  - File Sender Client program running on VM which transfers the VM log files from VM to PM via a direct communication channel.
  - File Sender daemon running on VM which regularly executes the File Sender Client program.
  - File Sender Server program running on host PMs which receives the VM log files sent by the File Sender Client program.
  - Two Database Loader daemons running on PM. The first one regularly loads the VM log files into a remote database server. The second one regularly loads the PM log files into the same remote database server.

With these components, we can then view and analyze the consolidated VM and PM flogs using any database front-end tools or in spreadsheet tools reading comma-separated value (CSV)/ tab-separated value (TSV) files.

2) How Flogger Works

Flogger captures file-centric logs (*a.k.a.* flogs) via the following steps (with reference to the labels in Figure 1):

Step 1: Linux Flogger/ Windows Flogger intercept every file access in the VMs. The Floggers capture the following information (Flog Subset A) (non-exhaustive list):

- VM Accessed file name and full path e.g. /home/users/john/docs/sensitive.txt
- VM File access date/time
- VM IP address
- VM MAC address
- Machine type i.e. VM or PM
- UID of file owner of the accessed file
- GID of file owner of the accessed file
- UID of process owner who accessed the file
- GID of process owner who accessed the file
- Action done to accessed file e.g. Create, Read, Write, Socket (Send Message), Socket (Receive Message), Delete

It is important to note that the list in Flog Subset A is not exhaustive and more attributes are added to make the system more robust, e.g. more timestamps.

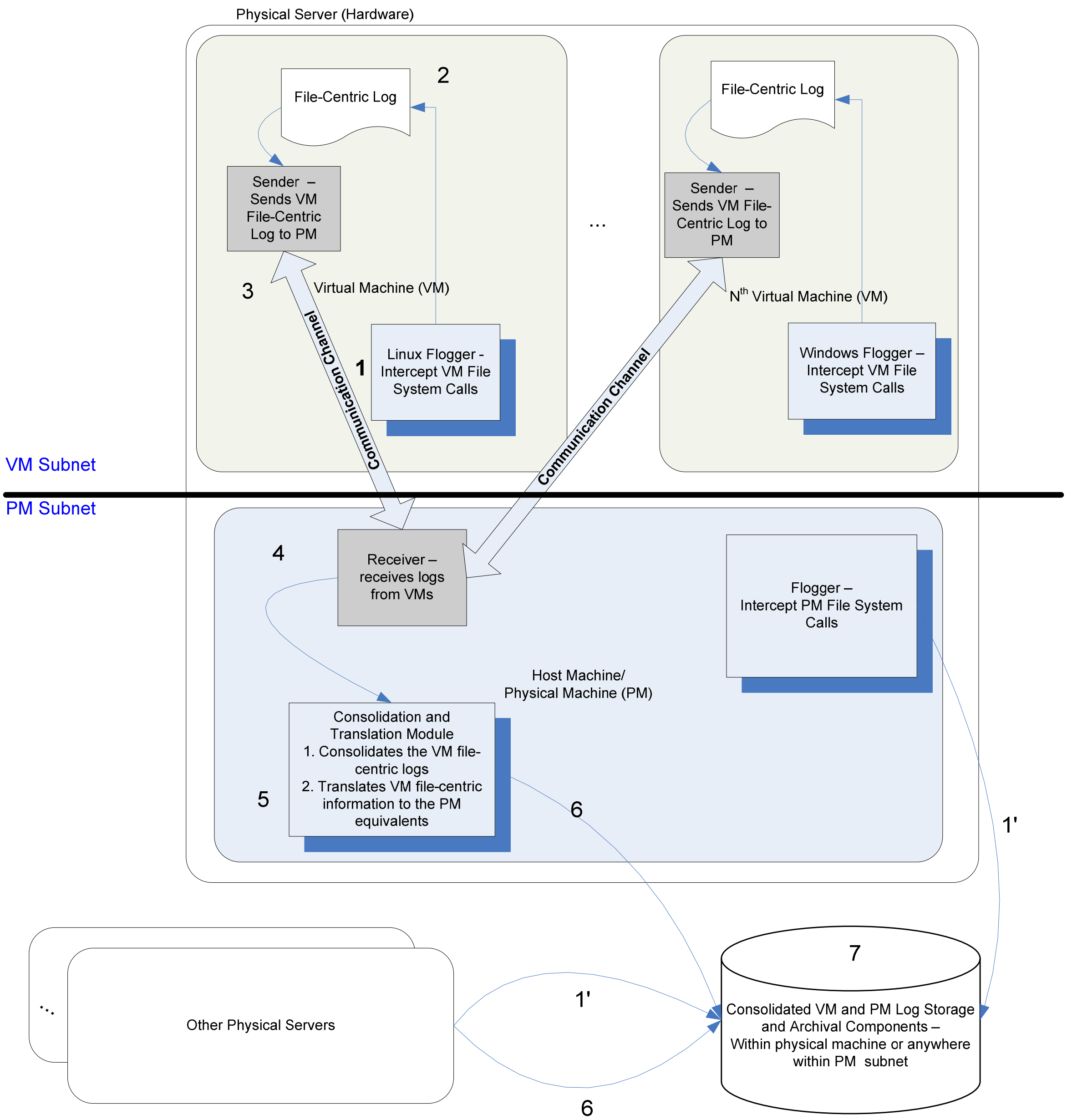

Figure 1. Architecture and flow of interactions and information passing between Flogger components

Output pane

Data Output | Explain | Messages | History

	filename text	full_path text	action text	date_time timestamp with time zone	file_username text	process_username text	vm_ip4 text
1	mounts	/3350/mounts	Read	2011-03-23 14:00:26+08	tomcat	tomcat	192.168.198.130
2	install.txt	/home/tomcat/kernel trust/install.txt	Read	2011-03-23 14:00:34+08	tomcat	tomcat	192.168.198.130
3	mounts	/3352/mounts	Read	2011-03-23 14:00:41+08	tomcat	tomcat	192.168.198.130
4	document.txt	/home/tomcat/kernel trust/test/document.txt	Create	2011-03-23 14:00:53+08	tomcat	tomcat	192.168.198.130
5	mounts	/3356/mounts	Read	2011-03-23 14:01:01+08	tomcat	tomcat	192.168.198.130
6	listing.txt	/home/tomcat/kernel trust/listing.txt	Read	2011-03-23 14:01:14+08	tomcat	tomcat	192.168.198.130
7	document.txt	/home/tomcat/kernel trust/test/document.txt	Read	2011-03-23 14:01:24+08	tomcat	tomcat	192.168.198.130
8	document.txt	/home/tomcat/kernel trust/test/document.txt	Write	2011-03-23 14:01:28+08	tomcat	tomcat	192.168.198.130
9	mounts	/3369/mounts	Read	2011-03-23 14:01:31+08	tomcat	tomcat	192.168.198.130
10	document2.txt	/home/tomcat/kernel trust/test/document2.txt	Create	2011-03-23 14:01:43+08	tomcat	tomcat	192.168.198.130
11	mounts	/3371/mounts	Read	2011-03-23 14:01:49+08	tomcat	tomcat	192.168.198.130
12	mounts	/3374/mounts	Read	2011-03-23 14:02:01+08	tomcat	tomcat	192.168.198.130
13	test.txt	/home/tomcat/kernel trust/src/test.txt	Read	2011-03-23 14:02:09+08	tomcat	tomcat	192.168.198.130
14	mounts	/3376/mounts	Read	2011-03-23 14:02:11+08	tomcat	tomcat	192.168.198.130
15	install.txt	/home/tomcat/kernel trust/install.txt	Read	2011-03-23 14:02:22+08	tomcat	tomcat	192.168.198.130
16	mounts	/3379/mounts	Read	2011-03-23 14:02:30+08	tomcat	tomcat	192.168.198.130
17	document.txt	/home/tomcat/kernel trust/test/document.txt	Read	2011-03-23 14:02:35+08	tomcat	tomcat	192.168.198.130
18	document.txt	/home/tomcat/kernel trust/test/document.txt	Write	2011-03-23 14:02:41+08	tomcat	tomcat	192.168.198.130
19	document2.txt	/home/tomcat/kernel trust/test/document2.txt	Read	2011-03-23 14:02:45+08	tomcat	tomcat	192.168.198.130
20	document2.txt	/home/tomcat/kernel trust/test/document2.txt	Write	2011-03-23 14:02:48+08	tomcat	tomcat	192.168.198.130
21	mounts	/3382/mounts	Read	2011-03-23 14:02:48+08	tomcat	tomcat	192.168.198.130
22	inaddress.c	/home/tomcat/kernel trust/test/inaddress.c	Read	2011-03-23 14:03:02+08	tomcat	tomcat	192.168.198.130
23	mounts	/3384/mounts	Read	2011-03-23 14:03:05+08	tomcat	tomcat	192.168.198.130
24	split.c	/home/tomcat/kernel trust/test/split.c	Read	2011-03-23 14:03:07+08	tomcat	tomcat	192.168.198.130
25	mounts	/3387/mounts	Read	2011-03-23 14:03:12+08	tomcat	tomcat	192.168.198.130
26	document.txt	/home/tomcat/kernel trust/test/document.txt	Delete	2011-03-23 14:03:18+08	tomcat	tomcat	192.168.198.130

	vm_ip6 text	vm_mac text	vm_interface text	pm_ip4 text	pm_mac text	pid integer	p_uid integer	p_gid integer	timeval_sec bigint	timeval_usec bigint
1	fe80000000000000020c29fffe5921a7	00:0c:29:59:21:a7	eth0	192.168.198.1	a0:7f:4d:b2:6d:c5	3350	500	500	1300860025	543880
2	fe80000000000000020c29fffe5921a7	00:0c:29:59:21:a7	eth0	192.168.198.1	a0:7f:4d:b2:6d:c5	3351	500	500	1300860033	219409
3	fe80000000000000020c29fffe5921a7	00:0c:29:59:21:a7	eth0	192.168.198.1	a0:7f:4d:b2:6d:c5	3352	500	500	1300860040	457650
4	fe80000000000000020c29fffe5921a7	00:0c:29:59:21:a7	eth0	192.168.198.1	a0:7f:4d:b2:6d:c5	3353	500	500	1300860053	136480
5	fe80000000000000020c29fffe5921a7	00:0c:29:59:21:a7	eth0	192.168.198.1	a0:7f:4d:b2:6d:c5	3356	500	500	1300860060	300614
6	fe80000000000000020c29fffe5921a7	00:0c:29:59:21:a7	eth0	192.168.198.1	a0:7f:4d:b2:6d:c5	3367	500	500	1300860073	505171
7	fe80000000000000020c29fffe5921a7	00:0c:29:59:21:a7	eth0	192.168.198.1	a0:7f:4d:b2:6d:c5	3368	500	500	1300860083	278415
8	fe80000000000000020c29fffe5921a7	00:0c:29:59:21:a7	eth0	192.168.198.1	a0:7f:4d:b2:6d:c5	3368	500	500	1300860087	485114
9	fe80000000000000020c29fffe5921a7	00:0c:29:59:21:a7	eth0	192.168.198.1	a0:7f:4d:b2:6d:c5	3369	500	500	1300860090	574643
10	fe80000000000000020c29fffe5921a7	00:0c:29:59:21:a7	eth0	192.168.198.1	a0:7f:4d:b2:6d:c5	3370	500	500	1300860102	566805
11	fe80000000000000020c29fffe5921a7	00:0c:29:59:21:a7	eth0	192.168.198.1	a0:7f:4d:b2:6d:c5	3371	500	500	1300860108	933135
12	fe80000000000000020c29fffe5921a7	00:0c:29:59:21:a7	eth0	192.168.198.1	a0:7f:4d:b2:6d:c5	3374	500	500	1300860120	786127
13	fe80000000000000020c29fffe5921a7	00:0c:29:59:21:a7	eth0	192.168.198.1	a0:7f:4d:b2:6d:c5	3375	500	500	1300860128	272957
14	fe80000000000000020c29fffe5921a7	00:0c:29:59:21:a7	eth0	192.168.198.1	a0:7f:4d:b2:6d:c5	3376	500	500	1300860130	700398
15	fe80000000000000020c29fffe5921a7	00:0c:29:59:21:a7	eth0	192.168.198.1	a0:7f:4d:b2:6d:c5	3377	500	500	1300860142	50178
16	fe80000000000000020c29fffe5921a7	00:0c:29:59:21:a7	eth0	192.168.198.1	a0:7f:4d:b2:6d:c5	3379	500	500	1300860149	806354
17	fe80000000000000020c29fffe5921a7	00:0c:29:59:21:a7	eth0	192.168.198.1	a0:7f:4d:b2:6d:c5	3380	500	500	1300860155	123242
18	fe80000000000000020c29fffe5921a7	00:0c:29:59:21:a7	eth0	192.168.198.1	a0:7f:4d:b2:6d:c5	3380	500	500	1300860161	11626
19	fe80000000000000020c29fffe5921a7	00:0c:29:59:21:a7	eth0	192.168.198.1	a0:7f:4d:b2:6d:c5	3381	500	500	1300860164	678920
20	fe80000000000000020c29fffe5921a7	00:0c:29:59:21:a7	eth0	192.168.198.1	a0:7f:4d:b2:6d:c5	3381	500	500	1300860167	232444
21	fe80000000000000020c29fffe5921a7	00:0c:29:59:21:a7	eth0	192.168.198.1	a0:7f:4d:b2:6d:c5	3382	500	500	1300860168	70155
22	fe80000000000000020c29fffe5921a7	00:0c:29:59:21:a7	eth0	192.168.198.1	a0:7f:4d:b2:6d:c5	3383	500	500	1300860181	358884
23	fe80000000000000020c29fffe5921a7	00:0c:29:59:21:a7	eth0	192.168.198.1	a0:7f:4d:b2:6d:c5	3384	500	500	1300860184	984029
24	fe80000000000000020c29fffe5921a7	00:0c:29:59:21:a7	eth0	192.168.198.1	a0:7f:4d:b2:6d:c5	3385	500	500	1300860186	965535
25	fe80000000000000020c29fffe5921a7	00:0c:29:59:21:a7	eth0	192.168.198.1	a0:7f:4d:b2:6d:c5	3387	500	500	1300860191	740271
26	fe80000000000000020c29fffe5921a7	00:0c:29:59:21:a7	eth0	192.168.198.1	a0:7f:4d:b2:6d:c5	3388	500	500	1300860197	253185

Figure 2. Sample consolidated file-centric log (flog) extracted from querying the log storage

Table 1: Extracted columns from flogs of Scenario 1

No. (Included for this paper)	filename	full_path	uid	gid	file_username	pid	p_uid	process_username	vm_ip4	vm_ip6	vm_mac	vm_interface	date_time	timeval_sec	timeval_usec	vm_ip4_raw	action	…
1	PatentDisclosure.txt	/shared/doc/PatentDisclosure.txt	502	502	alice	24436	502	alice	10.252.250.1	fe800000000000020c29fffec5bc44	00:0c:29:c5:bc:44	eth0	2011-06-27 09:55:24+08	1309139727	738618	184351233	Create	…
2	PatentDisclosure.txt	/shared/doc/PatentDisclosure.txt	502	503	alice	24436	502	alice	10.252.250.1	fe800000000000020c29fffec5bc44	00:0c:29:c5:bc:44	eth0	2011-06-27 09:55:24+08	1309139727	739308	184351233	Read	…
3	PatentDisclosure.txt	/shared/doc/PatentDisclosure.txt	502	503	alice	24524	502	alice	10.252.250.1	fe800000000000020c29fffec5bc44	00:0c:29:c5:bc:44	eth0	2011-06-27 09:56:47+08	1309139810	672980	184351233	Read	…
4	PatentDisclosure.txt	/shared/doc/PatentDisclosure.txt	502	503	alice	24524	502	alice	10.252.250.1	fe800000000000020c29fffec5bc44	00:0c:29:c5:bc:44	eth0	2011-06-27 09:57:00+08	1309139823	808734	184351233	Write	…
5	PatentDisclosure.txt	/shared/doc/PatentDisclosure.txt	502	503	alice	24524	502	alice	10.252.250.1	fe800000000000020c29fffec5bc44	00:0c:29:c5:bc:44	eth0	2011-06-27 09:57:00+08	1309139823	836413	184351233	Read	…
6	PatentDisclosure.txt	/shared/doc/PatentDisclosure.txt	502	503	alice	24524	502	alice	10.252.250.1	fe800000000000020c29fffec5bc44	00:0c:29:c5:bc:44	eth0	2011-06-27 09:57:00+08	1309139823	837186	184351233	Rename (Old File)	…
7	PatentDisclosure.txt~	/shared/doc/PatentDisclosure.txt~	502	502	alice	24524	502	alice	10.252.250.1	fe800000000000020c29fffec5bc44	00:0c:29:c5:bc:44	eth0	2011-06-27 09:57:00+08	1309139823	837735	184351233	Rename (New File)	…
8	PatentDisclosure.txt	/shared/doc/PatentDisclosure.txt	502	502	alice	24524	502	alice	10.252.250.1	fe800000000000020c29fffec5bc44	00:0c:29:c5:bc:44	eth0	2011-06-27 09:57:00+08	1309139823	841338	184351233	Rename (New File)	…
9	PatentDisclosure.txt	/shared/doc/PatentDisclosure.txt	502	503	alice	24524	502	alice	10.252.250.1	fe800000000000020c29fffec5bc44	00:0c:29:c5:bc:44	eth0	2011-06-27 09:57:00+08	1309139823	844019	184351233	Read	…
10	PatentDisclosure.txt	/shared/doc/PatentDisclosure.txt	502	503	alice	24590	502	alice	10.252.250.1	fe800000000000020c29fffec5bc44	00:0c:29:c5:bc:44	eth0	2011-06-27 09:58:34+08	1309139917	782164	184351233	Read	…
11	PatentDisclosure.txt	/shared/doc/PatentDisclosure.txt	502	503	alice	24595	502	alice	10.252.250.1	fe800000000000020c29fffec5bc44	00:0c:29:c5:bc:44	eth0	2011-06-27 09:58:50+08	1309139933	983277	184351233	Read	…
12	PatentDisclosure.txt	/shared/doc/PatentDisclosure.txt	502	503	alice	24631	503	bob	10.252.250.1	fe800000000000020c29fffec5bc44	00:0c:29:c5:bc:44	eth0	2011-06-27 10:00:34+08	1309140037	509728	184351233	Read	…

Step 1': Just like VMs, PMs also have Floggers which intercept the PMs file system calls and then stores them in the Data Store.

Step 2: After the file life-cycle related information are captured, they are sent to the host PM. The VM Flogger directly sends the captured information (Flog Subset A) to PM Receiver Daemon via a Communication Channel between VM and PM. The Communication Channel is special mechanism available on typical hypervisors which enable a serial cable-like communication between VMs and PMs. It does not involve networking transfers. Hence, no VM Flogger transfer Flogs to PM File Sender Servers via network transfers. This increases the security of the transfer of Flogs.

Step 3: VM File Sender Daemon regularly executes the File Sender Client which reads the File Access Details (Flog Subset A) and sends them to the PM via the Communication Channel between VM and PM.

Step 4: PM File Sender Server receives the File Access Details (Flog Subset A) from VM File Sender Client via the Communication Channel between VM and PM.

Step 5: PM Flogger generates other PM information (Flog Subset B), for example (but not limited to):

- PM IP address
- PM MAC address

Step 6: The PM Flogger sends Subset B to PM File Sender Server. Subsets A & B will give users a consolidated set of information (i.e. Flog) which can pinpoint the VMs and PMs involved in each file's life cycle to enable full accountability of distributed VM and PM architectures, e.g. Cloud computing.

Step 7: Within the PM Subnet, the PM Database loader daemons write the joint/ consolidated information (both Subset A & Subset B) to a Data Store e.g. database for future data mining and reporting. Note that all the consolidation of the Flogs across PMs into the Data Store take place only in the PM Subnet. Users in the VM Subnet should have no awareness of these behind-the-scenes steps. It is also noteworthy to know that we have not decided on the exact short, medium and long term storage of flogs, as this require another set of I/O experiments against benchmarks and scale.

V. RESULTS AND EXAMPLE SCENARIOS

This paper reports our initial experiments focusing on deploying Floggers to capture flogs across VMs and PMs for a Cloud, and also to demonstrate that we are able to join the information for VMs and their underlying host PMs. This gives a comprehensive overview of the file-centric accesses and transfers within a typical Cloud. Many other research topics and questions were raised and they will be covered in Section VI.

A. Environments Experiments Conducted In

In order to prove the concept of Flogger, we have developed and run the implementation of Flogger on the following operating systems:

- Flogger (Linux) in the Linux Family
  - CentOS 5.3
  - Fedora 15
  - Ubuntu 11.04
- Flogger (Windows) in the Windows Family
  - Windows XP Professional SP3
  - Windows Server 2008 R2

Flogs generated were also pushed into databases via the DB loaders. Experiments were conducted against the prominent open-source row-based relational database PostgreSQL 9.0 and in preparation for data analytical needs over flogs, we also experimented with the column-store MonetDB.

B. Use Case Scenarios

To illustrate the Flogger in action, we will explain two example scenarios. It is important to note the number of scenarios is not exhaustive, and they serve to enhance the appreciation of the usage and potential of Flogger.

1) Example Scenario 1: Recording and Detection of Unauthorized User Accessing a File

In this scenario, a fictitious user 'Alice' creates a sensitive document (PatentDisclosure.txt) and modified the document. Some time later, another user 'Bob' reads the sensitive document without Alice's permission.

Table 1 shows a subset of the columns and results of flogs from a VM as a result of enacting this scenario. The log rows number 1 to 11, excluding 6 to 8, depict Alice creating and modifying the sensitive document. The log rows number 6 to 8 (the Rename operations) depict the text editor doing some behind-the-scene housekeeping operations during a save operation. Interestingly, the log row number 12 depicts Bob reading the sensitive document without Alice's permission. Note that Bob's username is displayed in row 12 instead of Alice's username.

2) Example Scenario 2: Capturing of File Transfers Across VMs in the Cloud

In the next scenario, we show Flogger capturing file transfers within the Cloud. The first VM running CentOS 5.3 sends a file (testcopy.txt) via the Linux program *scp* (Secure Copy) to the second VM running Ubuntu 11.04.

In Figures 3(a) to 3(d), the sender VM log rows number 2 and 7 depict the network operations (Socket (Send Message)) when the first VM is sending the file. (We have split up the table into parts *a* to *d* due to space reasons).

	A	B	C	D	E
1	Filename	Full_Path	UID	GID	File_Username
2	N/A Socket	N/A Socket	0	0	root
3	mounts	/3297/mounts	500	500	tomcat
4	mounts	/3298/mounts	500	500	tomcat
5	known_hosts	/home/tomcat/.ssh/known_hosts	500	500	tomcat
6	testcopy.txt	/home/tomcat/testcopy.txt	500	500	tomcat
7	N/A Socket	N/A Socket	0	0	root

(a)

	F	G	H	I	J	K	L	M	N
1	PID	P_UID	P_EUID	P_SUID	P_FSUID	P_GID	P_EGID	P_SGID	P_FSGID
2	2982	500	500	500	500	500	500	500	500
3	3297	500	500	500	500	500	500	500	500
4	3298	500	500	500	500	500	500	500	500
5	3298	500	500	500	500	500	500	500	500
6	3297	500	500	500	500	500	500	500	500
7	3233	0	0	0	0	0	0	0	0

(b)

	O	P	Q	R	S
1	Process_Username	VM_IP4	VM_IP6	VM_MAC	VM_Interface
2	tomcat	192.168.198.130	fe8000000000000020c29fffe5921a7	00:0c:29:59:21:a7	eth0
3	tomcat	192.168.198.130	fe8000000000000020c29fffe5921a7	00:0c:29:59:21:a7	eth0
4	tomcat	192.168.198.130	fe8000000000000020c29fffe5921a7	00:0c:29:59:21:a7	eth0
5	tomcat	192.168.198.130	fe8000000000000020c29fffe5921a7	00:0c:29:59:21:a7	eth0
6	tomcat	192.168.198.130	fe8000000000000020c29fffe5921a7	00:0c:29:59:21:a7	eth0
7	root	192.168.198.130	fe8000000000000020c29fffe5921a7	00:0c:29:59:21:a7	eth0

(c)

	T	U	V	W	X
1	Date_Time	Timeval_sec	Timeval_usec	VM_IP4_Raw	Action
2	7/19/2011 11:14	1311045263	92778	3232286338	Socket (Send Message)
3	7/19/2011 11:14	1311045263	97762	3232286338	Read
4	7/19/2011 11:14	1311045263	140760	3232286338	Read
5	7/19/2011 11:14	1311045263	274018	3232286338	Read
6	7/19/2011 11:14	1311045266	117411	3232286338	Read
7	7/19/2011 11:14	1311045273	398189	3232286338	Socket (Send Message)

(d)

Figure 3. Extract from Scenario 2 Sender Logs

	A	B	C	D	E
1	Filename	Full_Path	UID	GID	File_Username
123	testcopy.txt	/home/tomcat/testcopy.txt	1000	1000	tomcat
124	N/A Socket	N/A Socket	0	0	root

(a)

	F	G	H	I	J	K	L	M	N
1	PID	P_UID	P_EUID	P_SUID	P_FSUID	P_GID	P_EGID	P_SGID	P_FSGID
123	2467	1000	1000	1000	1000	1000	1000	1000	1000
124	472	0	0	0	0	0	0	0	0

(b)

	O	P	Q	R	S
1	Process_Username	VM_IP4	VM_IP6	VM_MAC	VM_Interface
123	tomcat	192.168.198.134	fe8000000000000020c29fffec1c1dc	00:0c:29:c1:c1:dc	eth0
124	root	192.168.198.134	fe8000000000000020c29fffec1c1dc	00:0c:29:c1:c1:dc	eth0

(c)

	T	U	V	W	X
1	Date_Time	Timeval_sec	Timeval_usec	VM_IP4_Raw	Action
123	7/19/2011 3:14	1311045259	889785	3232286342	Write
124	7/19/2011 3:14	1311045259	944838	3232286342	Socket (Receive Message)

(d)

Figure 4. Extract from Scenario 2 Receiver Logs

In Figures 4(a) to 4(d), the receiver VM log row number 124 depicts one of the network operations (Socket (Receive Message)) when the second VM is receiving the file. Note: most of the receiver log rows have been hidden due to space constraint. Note that at the same time, corresponding physical machine logs are also generated in their underlying PMs during the scp transfer. Both sets of VM and PM logs can then be joined for further analysis and forensics.

Figures 5 and 6 below depict the Socket (Send Message) and Socket (Receive Message) respectively being captured in the Linux kernel message log.

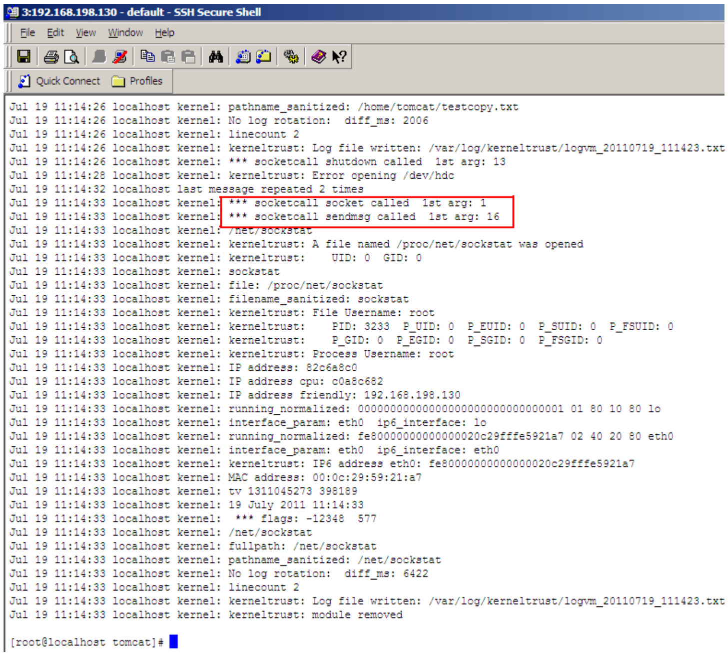

Figure 5. Scenario 2 Sender Kernel Message Log File

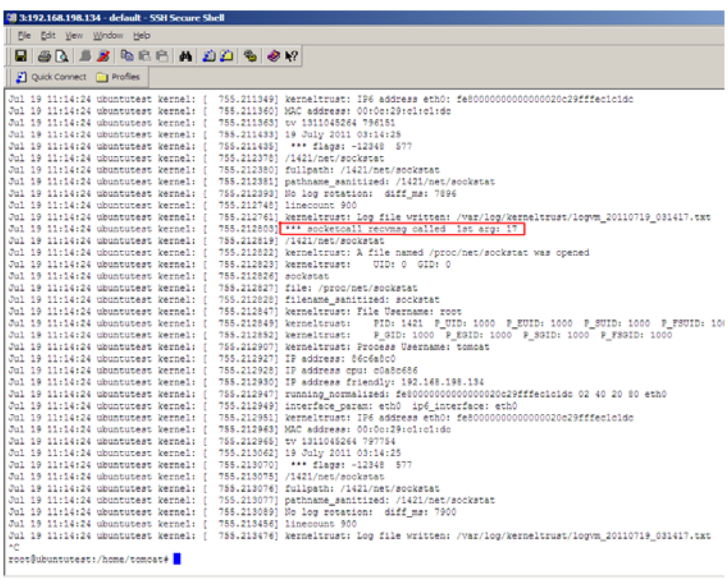

Figure 6. Scenario 2 Receiver Kernel Message Log File

C. *Provenance from Logs*

From the two scenarios, we can now visualize the data provenance potential information that Floggers can provide for Cloud end-users, administrators and even regulators. Virtual machine file access and transfers are logged with their corresponding file system calls in the physical hosts. Such correlation gives a good transparency of the location of files within a Cloud, and analytical tools can be built over these flogs to let people answer questions such as “Are my files really deleted in this Cloud?” or “Can I see who has accessed my sensitive file in this Cloud?”.

VI. Current and Future Work

The development of Floggers and the successful consolidation of simultaneously-generated VM and PM file-centric logs addressed the need for higher Cloud accountability and transparency, but also revealed limitations and several compelling future research directions:

A. *Integrity and Security of the Logger and Logs*

At the moment, flogs are passed securely down the communication channels from the VMs to their host PMs. As such, there is no network transfer of flogs at the virtual layer and the VM subnet (see Figure 1). Flogs consolidated at the PMs are sent to the data store within the PM subnet. Security of the Floggers also depends on the integrity of the machine kernels in the Cloud. However, the assumption of the kernel integrity is insufficient. Vulnerabilities may exist when PMs are transferring logs to the database storage. Authentication or simple client puzzles-like protocols between PMs and the storage may be introduced when flogs are transferred. There is also a need for flogs to remain tamper-proof and immutable. These requirements are our current top priorities.

B. *Scale and Log Data Size Explosion*

Compared to system-centric logs (e.g. event logs, system logs, or user account activity logs), file-centric logs (flogs) grow at a relatively higher rate. In one of our experiments, a file created in a word processing application generated up to approximately 29,000 file activities within 30 minutes even though user-triggered activities (e.g. write) are kept to the minimum. It was later revealed that its automatic backup features was enabled, causing it to be extremely chatty.

We are also aware that the prospect of flogs outgrowing the size of the actual files to be tracked is a realistic one. However, the concerns of the exponential growth of logs may be mitigated by our current attempts in exploring tiered storage and archival [1], de-duplication and summarization techniques.

C. *Rules for Application Footprints Captured in Flogs*

In our experiment, we also note an interesting observation of recurring footprints for different types of software. This opens the possibility of creating heuristics and rules for identification of anomalies and attacks in the Cloud.

D. *Visualizations*

With the large amount of data collected, it is perhaps a good idea to formulate different types of useful exploratory and presentation visualizations for the discovery and presentation of notable trends and patterns in the flogs. Visualization needs for end-users, administrators and regulators are different. For example, Cloud service providers may only offer end-users knowledge about the high-level geography without revealing specific data centers locations. End-users can still know if their data has violated cross-geography policies of data transfers. On the other hand, regulators may be granted special access accounts to visualize and audit the compliance of full data flows within the Cloud.

E. Linkage with Governance, Regulation and Compliance (GRC) needs

With the data accesses and transfers logged by Flogger, automatic auditing and high availability of data flow information are now realistic futures in the Cloud. This is also inline with the vision of the Workflow Layer in the TrustCloud framework[1].

VII. Concluding Remarks

In this paper, we emphasized the importance of a file-centric detective measures for increasing trust in the Cloud. We also demonstrated the increase of transparency and accountability of the Cloud via the novel file-centric logging mechanisms known as Floggers.

Current system logs only focuses on general system health indicators (e.g. uptimes, processor usage, events, etc). There is no focus on the life cycles of files stored in the file systems across both VM and PM. Our technique has addressed the need for a file-centric logging within networks of PMs hosting multiple-folds of VMs.

Moreover, current system logs are standalones kept within each VM or PM, and at best, across multiple VMs or PMs, but never consolidated or managed across both VM and PM simultaneously. There is a need for users to be aware of the exact VMs and the physical locations of underlying PMs that they have stored data in. Our technique, Flogger, has addressed this by logging file life-cycle related events on both VMs and their underlying host PMs.

Floggers can be applied into both private and public Cloud computing environments. Because of the service-oriented nature of Cloud services, Cloud users no longer need to own and maintain their own PMs, but rather, store their information in the Cloud, without the need to be concerned of the vendors' server system health indicators.

Our technique will enable system administrators and end-users to audit file life cycles, access and transfer histories. File-centric logs, or flogs, collected by Floggers will also enable system administrators and end-users to identify both the virtual and physical location of original and duplicate files to facilitate accountability, IT forensics and tracking of criminal activities within a Cloud provider's servers.

The initial experiments show a lot of promise. While there is much future work involved, we strongly feel that this is the exciting beginning of the distributed VM/PM file-centric logging paradigm for Cloud computing.

Acknowledgments

We would like to thank our colleagues Markus Kirchberg, Teck Hooi Lim, Alan Tan, Chun Hui Suen, Ahmed Aneeth, Ahmed Rifau Rasheed, Siani Pearson, and Miranda Mowbray from the TrustCloud research project [1] for their valuable critique and feedback.

References

[1] R.K.L. Ko, P. Jagadpramana, M. Mowbray, S. Pearson, M. Kirchberg, Q. Liang and B.S. Lee, "TrustCloud - A Framework for Accountability and Trust in Cloud Computing," *Proc. IEEE 2nd Cloud Forum for Practitioners (IEEE ICFP 2011)*, IEEE Computer Society, 2011, pp. 1-5.

[2] R.K.L. Ko, B.S. Lee and S. Pearson, "Towards Achieving Accountability, Auditability and Trust in Cloud Computing," *Proc. International workshop on Cloud Computing: Architecture, Algorithms and Applications (CloudComp2011)*, Springer, 2011, pp. 5-18.

[3] A.J.G. Hey, S. Tansley and K.M. Tolle, *The fourth paradigm: data-intensive scientific discovery*, Microsoft Research Redmond, WA, 2009.

[4] S. Sakr, A. Liu, D. Batista and M. Alomari, "A Survey of Large Scale Data Management Approaches in Cloud Environments," *Communications Surveys & Tutorials, IEEE*, no. 99, pp. 1-26.

[5] R. Love, "Kernel Korner: Intro to iNotify," *Linux Journal*, vol. 2005, no. 139, 2005, pp. 8.

[6] S.E. Hansen and E.T. Atkins, "Automated system monitoring and notification with swatch," USENIX Association's Proceedings of the Seventh Systems Administration (LISA VII) Conference, 1993.

[7] Silicon Graphics International Corp, "File Alteration Monitor (FAM) Overview," 2009; http://oss.sgi.com/projects/fam/.

[8] M. Roesch, "Snort-lightweight intrusion detection for networks," *Proc. 13th Large Installation System Administration Conference (LISA)*, 1999, pp. 229–238.

[9] HyTrust, "HyTrust Appliance," 2010; http://www.hytrust.com/product/overview/.

[10] Fujitsu Research Institute, "Personal data in the cloud: A global survey of consumer attitudes," 2010; http://www.fujitsu.com/downloads/SOL/fai/reports/fujitsu_personal-data-in-the-cloud.pdf.

[11] VMWare Hyperic, "Performance Monitoring for Cloud Services," 2011; http://www.hyperic.com/products/cloud-status-monitoring.

[12] CloudKick, "CloudKick - Cloud Monitoring and Management," 2011; https://www.cloudkick.com/.

[13] Cloud Security Alliance, "Security Guidance for Critical Areas of Focus in Cloud Computing V2.16," 2009; https://cloudsecurityalliance.org/csaguide.pdf.