Energy consumption side-channel attack at Virtual Machines in a Cloud

Helmut Hlavacs, Thomas Treutner
Research Group Entertainment Computing
University of Vienna, Austria
firstname.lastname@univie.ac.at

Jean-Patrick Gelas, Laurent Lefèvre, Anne-Cécile Orgerie
INRIA, LIP, Ecole Normale Supérieure de Lyon
University of Lyon, France
{jean-patrick.gelas, laurent.lefevre, annececile.orgerie}@ens-lyon.fr

***Abstract*—Virtualized data centers where several virtual machines (VMs) are hosted per server are becoming more popular due to Cloud Computing. As a consequence of energy efficiency concerns, the exact combination of VMs running on a specific server will most likely change over time. We present experimental results how to use the energy/power consumption logs of a power monitored server as a side-channel that allows us to recognize the exact combination of VMs it currently hosts to a high degree. For classification, we use a maximum log-likelihood approach, which works well for comparably small training and test set sizes. We also show to which degree a specific VM can be recognized, regardless of other VMs currently running on the same server, and show false negative/positive rates. To cross-validate our results, we have used a Kolmogorov-Smirnov test, resulting in comparable quality of recognition within shorter time. In order to clarify whether our approach is generalizable and yields reproducible results, we have set up a second experimental infrastructure in Lyon, using a different hardware platform and power measurement device. We have obtained similar results and have experimented with different CPU frequency scaling governors, yielding comparable quality of recognition. As a result, energy consumption data of servers must be protected carefully, as it is potentially valuable information for an attacker trying to track down a VM to mount further attack steps.**

I. Introduction

Virtualized data centers where several virtual machines (VMs) are hosted per server are state of the art and are expected to become even more popular in the future with new computing paradigms as Cloud Computing. As a consequence of energy efficiency or load balancing concerns within Cloud Computing infrastructures, the exact combination of VMs running on a specific server will most likely change over time by using live migration [2], [3]. Using virtualization often comes with the illusion of perfect isolation and therefore high privacy and security. There are many examples that such claims are in fact mostly illusionary. In contrast to direct attacks aiming at the software stack (e.g., the virtualization layer), we concentrate on indirect attacks, in this case using energy consumption logs. Such data is likely to exist in Cloud Computing infrastructures to be able to monitor the infrastructure state and to compute energy efficient workload mappings. In the future, it could be also used for accounting, as energy becomes a precious and expensive resource.

The aim of this work is to investigate the potential of extracting valuable information from raw energy consumption logs with impact on user privacy and security. We present experimental results, showing that it is indeed possible to track down the server hosting a VM in question, by using the servers power footprint. We regard our work as an additional, indirect attack vector, to gain more information about a most probably complex computing infrastructure, usable in a preparation stage before launching actual attack steps. E.g., even if the VM an attacker is aiming at is highly secured with a packet filter, intrusion detection system etc., all of such protection measures are defending the VM against attacks from the network or Internet uplink. Nevertheless, the VM is still completely dependent on the hypervisor's integrity and users have to put trust into the hypervisor not being malicious or compromised. Technologies like vTPM [9] are still in its early days, and just encrypting the VM's file systems will most probably not be a reliable remedy, as the data have to be decrypted at some point in time for computational tasks, unless of course the VM is exclusively used as a storage server for encrypted data. Then, the decrypted data will reside at least temporarily in the server's main memory, accessible by the hypervisor.

As a consequence, if the VM in question is not an adequate or convenient target for direct attacks because of whatever reason, there is still the hypervisor. Once compromised, it would allow privileged and yet, for the VM's owner, hardly detectable access to the VM. The most dangerous attackers are those trying to leave no traces at all and stay undetected. So, if attacking a VM would cause too much attention (e.g., because of an IDS) and the hypervisors in a Cloud infrastructure are insufficiently secured or have known vulnerabilities, they will be a convenient prey. Nevertheless, the fact that there will be at least dozens of hypervisors and only one of them is hosting the VM in question is still an open issue. The more hypervisors are required to be cracked to gain access to the VM, the more likely it is that the attack is detected even before fully carried out. However, if the attacker can retrieve power

consumption data, e.g., because of an insufficiently secured SNMP access to a metered rack PDU, it could be possible to extract information about which server the VM is running on. The attacker would then concentrate on this single machine to cause only a minimum of attention.

The rest of the paper is structured as follows: in Section II, we compare our work to related efforts and give examples for possible combination of approaches in the future. In Section III, we describe our experimental platforms and methodology. For reasons of clarity, there are subsections for each of the two platforms we have carried out our experiments. In Section IV, we explain the methodology of the different evaluations we have conducted. In Section V, we show and interpret the results we have obtained in our experiments. In order to be clear, Section V is organized in subsections to resemble the respective experimental platforms. Finally, we draw conclusions from our results in Section VI and state plans for extending the status quo in Section VII.

II. Related Work

With the new paradigm of Cloud Computing becoming more popular, an increasing number of new vulnerabilities are discovered, bugs that could not be exploited in the past are suddenly exploitable, and whole classes of attacks (e.g., timing attacks, power consumption attacks, etc.) find new scenarios of application [1]. Indeed, the addition of the hypervisor layer increases the range of possible attacks compared to non-virtualized environments [4].

Kortchinsky [7] attacks the virtualization layer and describes how a vulnerability in a graphics driver makes it possible to break out of a VMware guest and become root at the hypervisor level. The exploit only works on virtualized systems and in way that may become familiar in the future: The bug allows to access memory beyond the graphic memory boundary. In a non-virtualized system, this is clearly a bug, but most probable no vulnerability as physically non-existent memory can not be read or written. In a virtualized system, graphic memory is just a region of main memory, so access beyond the graphic memory boundary becomes meaningful and dangerous. Such a kind of attack is a possible step after a malicious user has found the server a specific VM is hosted on.

Cloud environments should guarantee the security of both user data and application processing [11]. As physical resources are shared between the guest virtual machines, flaws in the isolation of VMs lead attackers to have access to data and applications belonging to other users. This kind of threat is based on a two-step attack: targeted VM localization and then the attack itself. The localization phase has been proved to be feasible by Ristenpart et al. [10]. They carry out an attack on an infrastructure level. They are able to get enough information about the internals of Amazon EC2 to deliberately place a VM on the same host as a target VM. Additionally, they discuss how this ability can be used to start cross-VM side-channel attacks. We think our work is complementary in a way that the approaches may be combined to a faster or more accurate attack.

Once the targeted VM is localized, attacks relying on flaws and vulnerabilities are possible. For example, Ormandy [8] found a bug in the hypervisor code concerning the emulated I/O ports that allow a malicious user to write to an arbitrary out-of-bounds memory location. Another vulnerability in the guest virtual device driver allowed a guest VM to crash the host computer and the VMs hosted by it[1]. Numerous similar vulnerabilities have been found [4], [8].

This paper present an approach where attackers can use monitoring data such as power consumption of physical resources to locate a given VM. To the best of our knowledge, no research work deals with the privacy of power monitoring data in Cloud environments. But, this issue has been studied in the research fields concerning smart grid [12] and home metering [5]. Customers are indeed aware of what frequent data collected by home smart meters can reveal about their appliance usage.

III. Experimental Details

In the following, we describe our experimental setups and methodology of measurements. In a first step, we have examined the feasibility of our approach in Vienna. In order to clarify whether our approach is generalizable and yields reproducible results, we have set up a second experimental infrastructure in Lyon, using a different hardware platform and power measurement device.

In both of our experimental infrastructures, we have used four different `qemu-kvm/Linux` VMs in our experiments:

- A `MySQL` database VM, denoted by `sql`, hosting a dump of the Latin Wikipedia for the following `wp` VM
- an `Apache` web server VM with `PHP5` and `MediaWiki`, denoted by `wp`
- an `FTP` server VM, denoted by `ftp`
- and a VM doing I/O operations using `bonnie++`, denoted by `bonnie`.

E.g., the string `sql_ftp_bonnie` denotes that all VMs except the `Apache` VM were running. The string `wp` denotes that only the `Apache` VM was running. During the training phase, we have measured the energy consumption of a server for all 15 possible combinations (one combination of four VMs, four combinations of three VMs, six combinations of two VMs, and four combinations of a single VM running) of VMs running while a client sent a fixed trace of workload. The traces for `wp` and `ftp` were changing in concurrency of client requests, `sql` was acting as a database tier for `wp`. For `bonnie`, the number of benchmark runs within an interval was changed to simulate a varying I/O load.

A. Vienna

In our setup in Vienna, we have run the VMs on an AMD Phenom II X4 955 (3.2 GHz) CPU with 8 GB RAM. The VM disk images where hosted on an additional node with two SSDs in software-RAID0, connected over Gbit Ethernet.

To measure the power consumption, we have used a Hioki 3334 wattmeter device that allows several thousands measurements per second and integrates them to deliver active power consumption (W) and respectively energy consumption (Wh)

[1] http://www.vmware.com/security/advisories/VMSA-2009-0006.html

results. Based on the energy consumption, we have calculated the average power consumption within an interval of 6 s.

Each combination was traced for 40 minutes of workload, so we have approximately 400 data samples for each combination. During the whole experiment, the `ondemand` governor of the Linux CPU frequency system (for AMD platforms, `powernow_k8` driver) was activated to scale the frequency of all cores to the actual demand. The PhenomII X4 CPU is able to scale the frequency on a per-core level, so all of its four cores can be set to a different frequency. Available frequencies are 800 MHz, 2.1 GHz, 2.5 GHz and of course 3.2 GHz.

B. Lyon

In order to clarify whether our approach is generalizable and yields reproducible results, we have set up a second experimental infrastructure in Lyon. The hardware platform was comprised of six identical servers with Dual-Socket Quadcore Intel Xeon E5506 (2.13 GHz) and 12 GB RAM. On one of these servers, the VM disk images where hosted with five SSDs in hardware-RAID0, connected over Gbit Ethernet. The remaining five servers, all of them were identical, were used to run the VM combinations in parallel to speed up measurements.

To measure the power consumption, we have used a OmegaWatt multi-channel energy consumption logger which was able to measure all of the servers simultaneously and independently. The OmegaWatt device has no ability to integrate values, so its precision is not comparable to the Hioki 3334 used in Vienna. On the other hand, devices similar to the OmegaWatt logger are much more likely to be found in data centers than high precision devices intended for lab use, so we consider this as a more realistic attack ground. We have polled the OmegaWatt device every second, so for approximately 40 minutes of workload, we have obtained approximately 2400 power consumption samples.

In contrast to the PhenomII X4 CPU that was used in Vienna, the Xeon E5506 CPUs are not able to scale frequencies on a per-core level. Here, all cores of a CPU (socket) are set to the same frequency by the driver (`acpi_cpufreq`). E.g., if the server is only slightly loaded and only one core is required to run at a higher frequency than the minimum, all of the socket's CPU cores (here, four) will be automatically set to the same frequency by the `ondemand` governor.

We have carried out our experiments with three different scaling governors:

1) `ondemand`, to be able to compare the results to those obtained in Vienna,
2) `performance`, where all cores are fixed to the default frequency and scaling is in effect deactivated, and
3) `powersave`, where all cores are fixed to the minimum frequency.

With the `powersave` and `performance` governors, the range of possible power consumption should be smaller, which could make a difference for our evaluations. The available frequencies when using a Xeon E5506 CPU with the `ondemand` governor are: 2128 MHz, 1995 MHz, 1862 MHz, 1729 MHz and 1596 MHz. Additional to the lack of per-core scaling support, the Xeon CPU has a much smaller scaling range ($\frac{f_{min}}{f_{max}} = 0.75$) compared to the PhenomII ($\frac{f_{min}}{f_{max}} = 0.25$). The total power consumption of a CPU is the sum of dynamic and static power [6], and typically denoted by the approximation

$$P = ACV^2 f + VI_{leak},$$

where A is the fraction of gates actively switching and C the capacitance of all gates. A smaller range of frequency will therefore possibly lead to smaller range of power consumption. As there are other metrics influencing the power consumption (especially V^2 is contributing heavily) and the CPU is not the only device causing power consumption, we were interested if a different CPU frequency scaling governor makes a significant difference in our scenario.

IV. Evaluation

For evaluation, all consumption traces were split into training and test sample sets, for different test sample set sizes ranging from 5 to 50 in steps of 5. To minimize the effect of outlier results, the training/test sample set split was randomized and repeated 1000 times for each test sample set size. For each of these runs, we had to compute the probability density functions from scratch for each VM combination. To recognize the origin of a test sample set and therefore the combination of VMs, we used a maximum log-likelihood approach. For a given test sample set, we computed the log-likelihood for each possible origin to identify the most probable origin. To consider different perspectives, we have evaluated four different match conditions:

1) **Exact match:** The recognized combination is exactly matching the true combination, so the state of all VMs was correctly recognized.
2) **Subset match:** The true combination is subset of the recognized combination, so all running VMs were correctly recognized, but there are false positives (VMs falsely recognized as running).
3) **Set match:** This evaluation mode gives the degree of correctness of the classification. As we have four VMs and each of them can be running or not, we can interpret both the true and the recognized state as a quadruplet of $\{yes, no\}$. The degree of correctness is then the intersection of both quadruplets and possible values range from (0 of 4) to (4 of 4). For plotting purposes, we have scaled the range to [0;1].
   Based on the set match evaluation, we have calculated the false positive/negative recognition rates. A false positive in this context is that a VM was recognized running but is in fact not, and a false negative that a VM that was in fact running was not recognized so.
4) **VM match:** The last evaluation mode resembles the perspective of an attacker trying to track down a specific VM. In such a case, false positives/negatives are not important as long as the true state of a specific VM is recognized correctly to identify the server a VM is running on quickly. Once this server is found, further attack steps could be mounted.

V. Results

A. Vienna

Figure 1 and Figure 2 show the power consumption density for each VM combination, split into two plots for better visibility. We can see in Figure 2 that the power consumption of all combinations containing the `Apache` VM (`wp`) is very similar, and that they differ heavily from all combinations that do **not** contain the `Apache` VM.

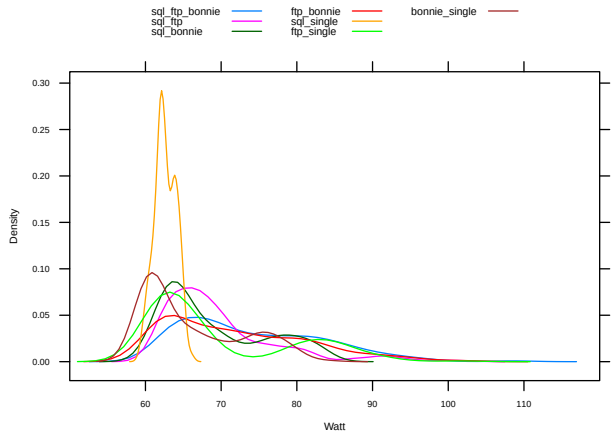

Fig. 1. Vienna, Probability density of power consumption, combinations **not** containing the `Apache` VM

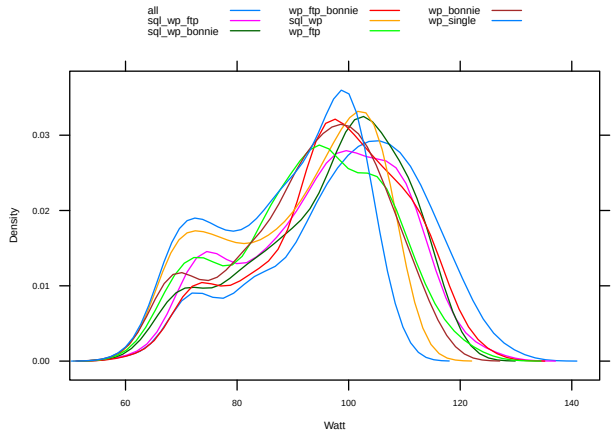

Fig. 2. Vienna, Probability density of power consumption, combinations **containing** the `Apache` VM

We can see in Figure 3 that for comparably small training and test set sizes, a lot of exact combinations can be recognized correctly to a high degree using the described log-likelihood approach. Most of the combinations yielding lower values of recognition rates have a running `wp` VM (`Apache, PHP`) in common.

We could see in Figure 2 that all combinations with `wp`

1) have a very similar power consumption trace and
2) have a higher average power consumption than all combinations without `wp`.

The consequences of these facts are that

1) it is hard to recognize which VMs are running in combination with `wp`, but

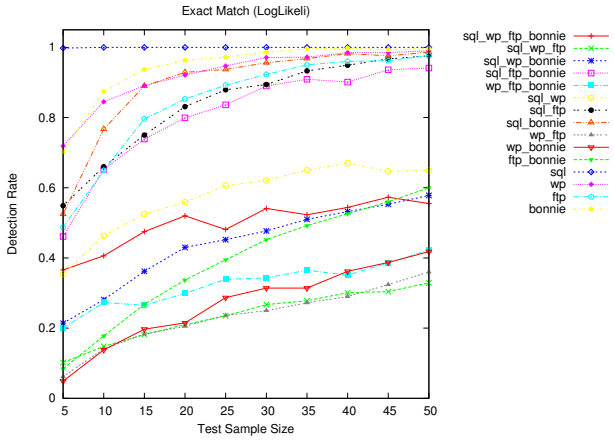

Fig. 3. Vienna, Exact Match Condition: Recognition Rates are very high for combinations **not** containing `wp`

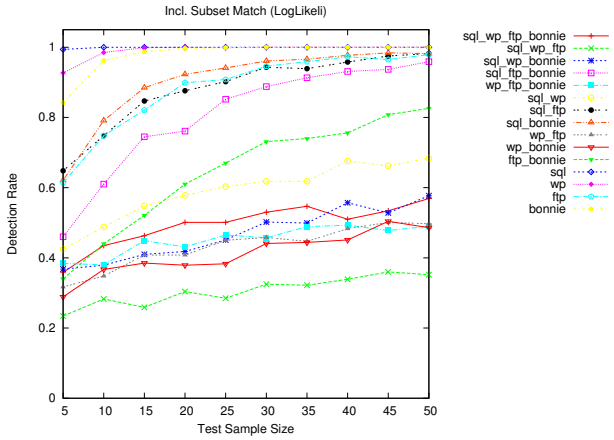

Fig. 4. Vienna, Subset Match Condition: Recognition Rates are generally increased

2) it is easy to recognize the state of `wp` correctly, which is clearly visible in Figure 6.

In Figure 4, we can see that if we relax the match condition from an exact match to a subset match, the match rate is increased, as false positives do not decrease the recognition rate, but only false negatives. For, e.g., the combinations `ftp_bonnie` and `wp_ftp`, the increase is clearly visible.

In Figure 5, the average degree of classification correctness is shown. Please note that the y-axis is scaled differently for better visibility. We see even higher recognition rates and that on average, the combinations with `wp` are classified falsely by a single VM that is not recognized correctly (3 of 4 correct).

Figure 6 shows the degree to which our approach is able to recognize the state of a specific VM correctly, regardless of the combination of VMs it is running with. This is perhaps the most interesting evaluation mode considering our scenario of privacy risks in a Cloud Computing infrastructure. Even for small numbers of test sample sizes, we can recognize whether a VM is running or not to a high degree.

In Figure 7, false positive and negative rates are visible and show how the respective gaps to a recognition rate of 1 in Figure 6 are composed by false positives and negatives. E.g.,

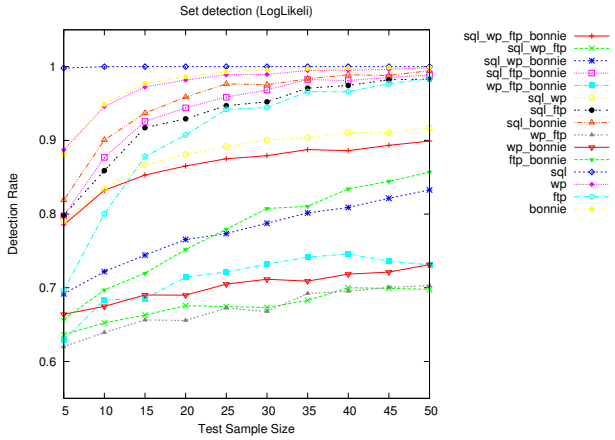

Fig. 5. Vienna, Set Match Condition: Average degree of combination recognition

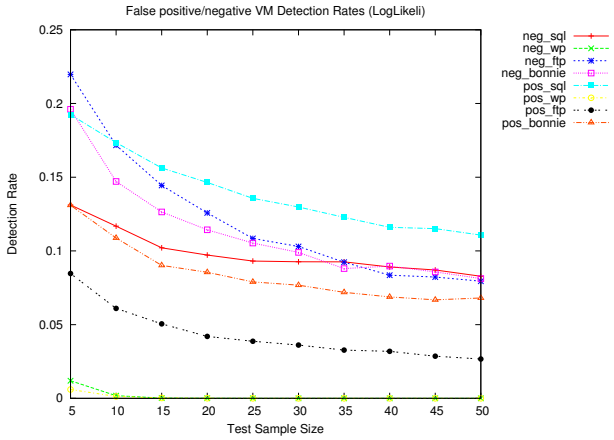

Fig. 7. Vienna, False Positive/Negative Recognition Rates

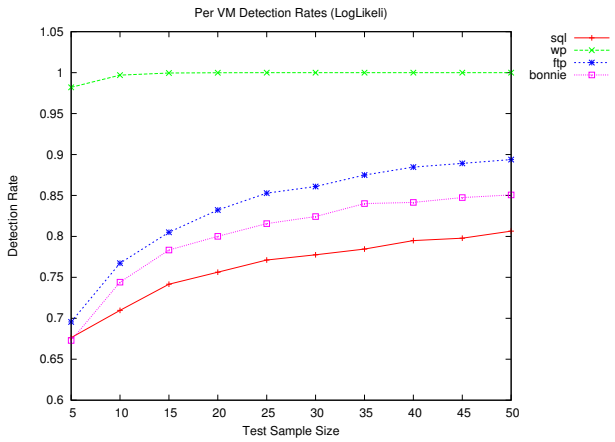

Fig. 6. Vienna, Per VM Recognition Rates: `wp` is extremely well recognizable

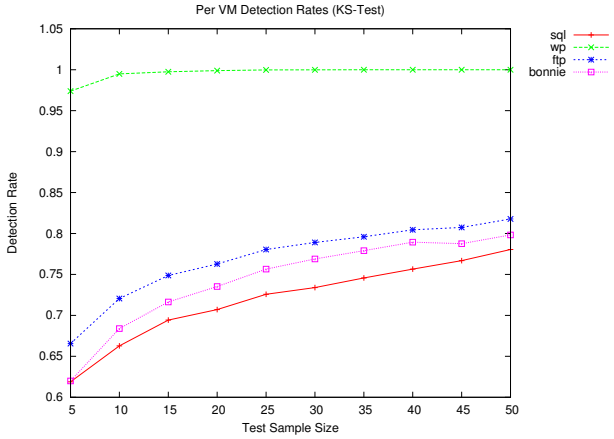

Fig. 8. Vienna, Per VM Recognition Rates using a Kolmogorov-Smirnov test for validation purposes, results are of comparable quality to Figure 6

we can see both high false positive and negative rates for `sql`, which is expectable due to the fact that `sql` has a very narrow density peak in a low region of power consumption. As a consequence, the weak `sql` signal will be hidden easily by additional, stronger signals, and it is hard to recognize the presence or absence of `sql` correctly. On the other hand, `wp`, the strongest of all signals, yields almost no false positives/negatives at all, but as mentioned above, the strong `wp` signal is hiding the weaker ones and makes it hard to recognize other VMs running in combination with `wp`.

Finally, we wanted to verify the described log-likelihood approach by using a Kolmogorov-Smirnov test for classification. Figure 8 shows the results for recognizing the state of a specific VM correctly, regardless of the combination of VMs it is running with. If we compare the results to Figure 6, we can see that the Kolmogorov-Smirnov test yields recognition rates approximately 5-10% lower than the log-likelihood approach. If we take into account that the log-likelihood approach requires frequent generation of density functions, which is a computationally highly intensive task and can easily take dozens of minutes for a fixed test sample set size and a sufficient number of repetitions, the quality of results obtained by using a much faster Kolmogorov-Smirnov test for classification could be acceptable too. Especially, if VMs are frequently live-migrated to other servers due to, e.g., load balancing or energy efficiency concerns, obtained results could be obsolete if the computation involved takes too long.

B. Lyon

In order to clarify whether our approach is generalizable and yields reproducible results, we have repeated and extended our experiments on a second infrastructure in Lyon, using a different hardware platform and power measurement device described in Section III.

In Figure 9, we can see the VM recognition rate when using the `ondemand` governor. The result should be compared to Figure 6, where the `ondemand` governor was used too, but on a different hardware platform etc., as described above. We can see that `wp` is achieving similarly high recognition rates. Interestingly, `ftp` is yielding slightly worse rates, but `sql` and `bonnie` significantly higher rates. If we compare the false positive and negative rates (Figure 7 and Figure 12), we can see that

- `ftp` has similar false negative rates, but much higher false positive rates,

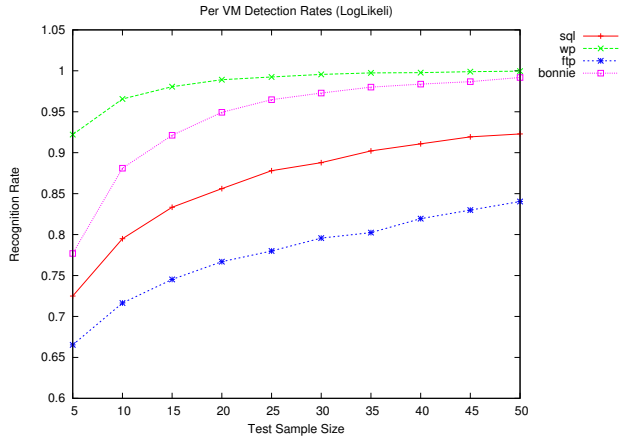

Fig. 9. Lyon, VM recognition rates for the `ondemand` governor scenario

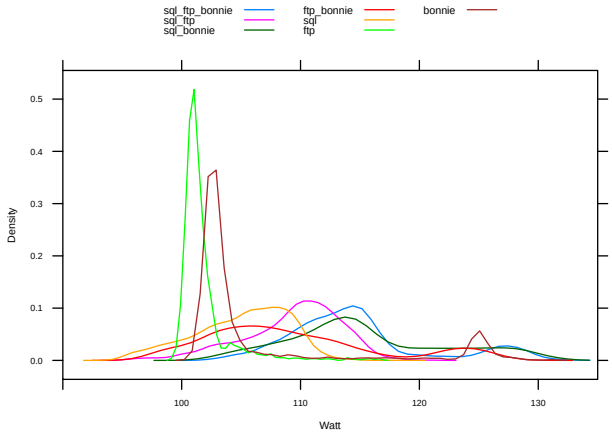

Fig. 10. Lyon, `ondemand` governor: Probability density of power consumption, combinations **not** containing the `Apache` VM

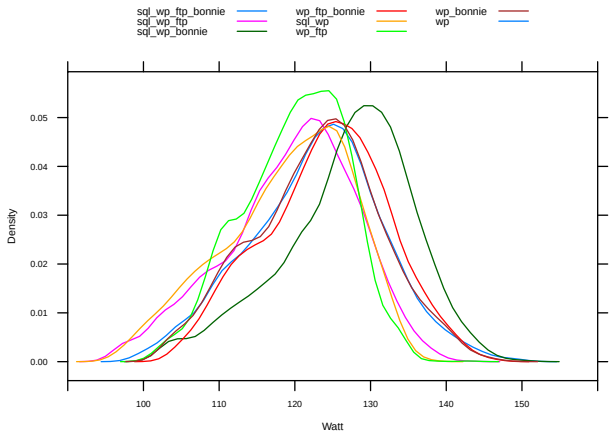

Fig. 11. Lyon, `ondemand` governor: Probability density of power consumption, combinations **containing** the `Apache` VM

- `sql` has similar false negative rates, but much lower false positive rates, and
- `bonnie` has much lower both false negative and positive rates.

Looking at the power consumption probability densities

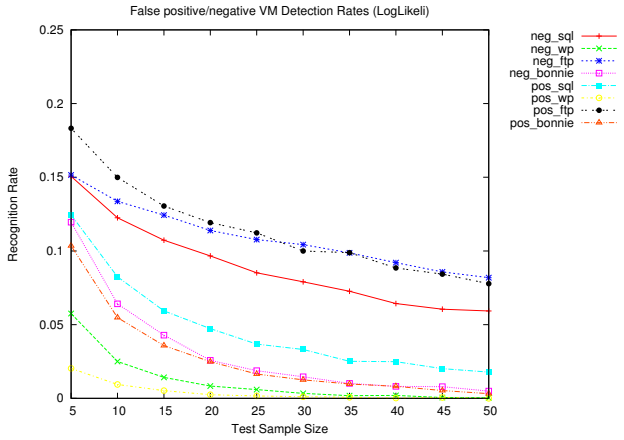

Fig. 12. Lyon, `ondemand` governor: False Positive/Negative Recognition Rates

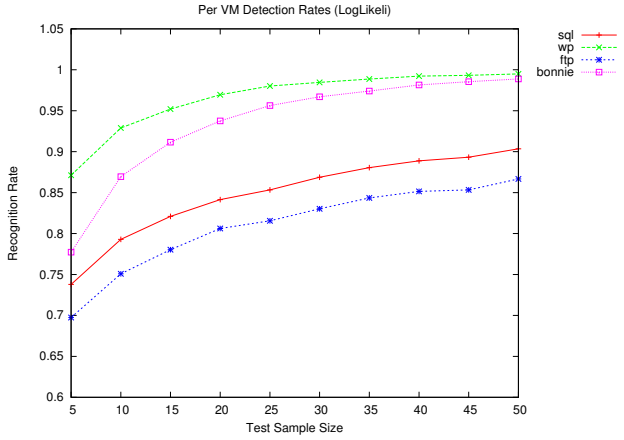

Fig. 13. Lyon, VM recognition rates for the `performance` governor scenario

when these VMs run alone (cf. Figure 10 and Figure 1) reveals that `ftp` is now the VM with the lowest and most narrow power footprint, and therefore the weakest signal, easily hidden by stronger ones if running in combination. The much higher false positive rate of `ftp` is a direct consequence. Vice versa, `sql` is not the weakest signal anymore, resulting in lower false positive rates. Finally, `bonnie` has a much more characteristic probability density when running alone, with two unique peaks. In consequence, all combinations containing `bonnie` are highly distinctive from all combinations not containing `bonnie`.

After we had empirical proof that our approach is generally working on a different hardware platforms too, we have investigated the potential impact of different CPU frequency scaling governors. In Figure 13, which should only be compared to Figure 9 and Figure 14, the results for the `performance` governor, where the frequency is fixed to the default frequency and scaling is in effect deactivated, are visible. The `sql` VM is yielding slightly worse rates compared to Figure 9, but despite that, no significant difference can be extracted.

Finally, in Figure 14, the results for a `powersave` governor scenario, where the frequency is fixed to the minimum

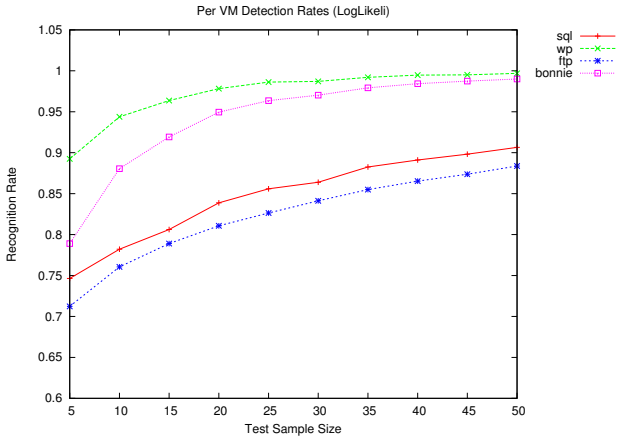

Fig. 14. Lyon, VM recognition rates for the `powersave` governor scenario

frequency, are shown and are not significantly differing from the `performance` scenario.

As a consequence, we can conclude that using the mentioned CPU frequency scaling governors the Linux kernel provides does not significantly influence the VM recognition rates and our approach is generalizable, including infrastructures using CPU frequency scaling due to energy efficiency concerns.

VI. Conclusion

We have presented experimental research on how to use the energy/power consumption logs of a power monitored server as a side-channel that allows us: a) to recognize the exact combination of VMs it currently hosts to a high degree, and b) to track down the server hosting a specific VM. The discussed results show that our approach is suitable to reduce the complexity of attacking a Cloud infrastructure when aiming at a specific VM, allowing the attacker to avoid being too conspicuous by having knowledge about the hypervisor a VM is running on.

Using the described maximum log-likelihood approach, a lot of exact combinations can be recognized correctly. If the match condition is relaxed to a subset match, the recognition rate is increased. To track down a specific VM, the state of the respective VM needs to be recognizable regardless of combination of VMs it may run in. Having similar consumption densities leads to a situation where it is hard to recognize the exact combination, but it is easy to recognize the VM that is causing the similarity.

We have compared the maximum log-likelihood approach against Kolmogorov-Smirnov method, yielding recognition rates approximately 5-10% lower than the maximum log-likelihood approach. Taking into account the much faster runtime of Kolmogorov-Smirnov and the fact that results can become obsolete due to live migration of VMs, we think that the quality of results obtained from using Kolmogorov-Smirnov could be acceptable too.

In order to clarify whether our approach is generalizable and yields reproducible results, we repeated and extended our experiments on a second infrastructure in Lyon, using a different hardware platform and a power measurement device likely to be found in data centers. We have obtained similar results and have experimented with different CPU frequency scaling governors, yielding comparable quality of recognition.

As a we have shown, power and energy consumption data of servers must be protected carefully, as it is possible to extract valuable information that an attacker could use to track down a VM before starting further attack steps. It is especially astonishing how few samples are required in our scenario for correct classification.

VII. Future Work

We plan to extend our work by adding random noise activity and therefore power consumption to investigate at which level of noise the recognition rate is suffering significantly. We also want to add more types of VMs and use different workload patterns too see if there is a level of complexity which is decreasing the recognition rate and respectively increasing the time required for computing reliable results. Additionally, there could be a point of complexity where Kolmogorov-Smirnov should be favored due to its lower runtime.

Finally, we plan to put effort into a more sophisticated attack scenario that does not require a training phase based on knowledge of the true combination of VMs running, but only information about the network address and kind of service (e.g., HTTP) of a VM, and access to an insufficiently secured metered rack PDU (e.g., by SNMP). Then, an attacker could send workload in varying intensities to the VM to cause changes in the power consumption of the server the VM is running on, trying to find the highest correlation between his actions and the consumption reactions. The assumption is that after removing noise, the server with the highest correlation in power consumption is, with high probability, the server hosting the VM in question. In such a scenario, the intensity and duration of injecting workload required to get reliable results is of high relevance, as a brute force approach would most probably rise suspicion.

VIII. Acknowledgements

The research leading to the results presented in this paper has been partly funded by EuroNF SJRP.44 SPEC project.

References

[1] Y. Chen, V. Paxson, and R. H. Katz. Whats new about cloud computing security? Technical Report UCB/EECS-2010-5, EECS Department, University of California, Berkeley, Jan 2010.

[2] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, pages 273–286, Berkeley, CA, USA, 2005. USENIX Association.

[3] F. Hermenier, X. Lorca, J.-M. Menaud, G. Muller, and J. Lawall. Entropy: a consolidation manager for clusters. In *VEE '09: Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pages 41–50, New York, NY, USA, 2009. ACM.

[4] W. Jansen and T. Grance. Guidelines on Security and Privacy in Public Cloud Computing. Draft from the National Institute of Standards and Technology, USA, 2011.

[5] G. Kalogridis, Z. Fan, and S. Basutkar. Affordable Privacy for Home Smart Meters. In *Workshop on Smart Grid Security and Communications (in conjunction with ISPA)*, pages 77–84, 2011.

[6] N. Kim, T. Austin, D. Baauw, T. Mudge, K. Flautner, J. Hu, M. Irwin, M. Kandemir, and V. Narayanan. Leakage current: Moore's law meets static power. *Computer*, 36(12):68–75, 2003.

[7] K. Kortchinsky. Cloudburst: Hacking 3d (and breaking out of vmware). In *Black Hat Technical Security Conference*, 2009.

[8] T. Ormandy. An Empirical Study into the Security Exposure to Hosts of Hostile Virtualized Environments. White Paper, 2008.

[9] R. Perez, R. Sailer, and L. van Doorn. vTPM: virtualizing the trusted platform module. In *Conference on USENIX Security Symposium*, 2006.

[10] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, pages 199–212, New York, NY, USA, 2009. ACM.

[11] N. Santos, K. Gummadi, and R. Rodrigues. Towards trusted cloud computing. In *Conference on Hot topics in cloud computing (HotCloud)*, Berkeley, CA, USA, 2009. USENIX Association.

[12] D. Seo, H. Lee, and A. Perrig. Secure and Efficient Capability-Based Power Management in the Smart Grid. In *Workshop on Smart Grid Security and Communications (in conjunction with ISPA)*, pages 119–126, 2011.