A Security Analysis of Amazon's Elastic Compute Cloud Service

Marco Balduzzi
EURECOM
marco.balduzzi@madlab.it

Jonas Zaddach
EURECOM
zaddach@eurecom.fr

Davide Balzarotti
EURECOM
balzarotti@eurecom.fr

Engin Kirda
Northeastern University
ek@ccs.neu.edu

Sergio Loureiro
SecludIT
sergio@secludit.com

ABSTRACT

Cloud services such as Amazon's Elastic Compute Cloud and IBM's SmartCloud are quickly changing the way organizations are dealing with IT infrastructures and are providing online services. Today, if an organization needs computing power, it can simply buy it online by instantiating a virtual server image on the cloud. Servers can be quickly launched and shut down via application programming interfaces, offering the user a greater flexibility compared to traditional server rooms.

This paper explores the general security risks associated with using virtual server images from the public catalogs of cloud service providers. In particular, we investigate in detail the security problems of public images that are available on the Amazon EC2 service. We describe the design and implementation of an automated system that we used to instantiate and analyze the security of public AMIs on the Amazon EC2 platform, and provide detailed descriptions of the security tests that we performed on each image. Our findings demonstrate that both the users and the providers of public AMIs may be vulnerable to security risks such as unauthorized access, malware infections, and loss of sensitive information. The Amazon Web Services Security Team has acknowledged our findings, and has already taken steps to properly address all the security risks we present in this paper.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms

Security, Design, Experimentation

Keywords

Cloud Computing, Elastic Compute Cloud Service, Security, AMI, Amazon

SAC'12 March 25-29, 2012, Riva del Garda, Italy.

1 Introduction

Cloud computing has changed the view on IT as a pre-paid asset to a pay-as-you-go service. Several companies such as *Amazon Elastic Compute Cloud*[4] *(EC2)*, *Rackspace*[5], *IBM SmartCloud*[8], *Joyent Smart Data Center*[10] or *Terremark vCloud*[6] are offering access to virtualized servers in their data centers on an hourly basis. Servers can be quickly launched and shut down via application programming interfaces, offering the user a greater flexibility compared to traditional server rooms. This paradigm shift is changing the existing IT infrastructures of organizations, allowing smaller companies that cannot afford a large infrastructure to create and maintain online services with ease.

A popular approach in cloud-based services is to allow users to create and share virtual images with other users. For example, a user who has created a legacy Linux Debian Sarge image may decide to make this image public so that other users can easily reuse it. In addition to user-shared images, the cloud service provider may also provide customized public images based on common needs of their customers (e.g., an Ubuntu web server image that has been pre-configured with MySQL, PHP and an Apache). This allows the customers to simply instantiate and start new servers, without the hassle of installing new software themselves.

In this paper, we explore the general security risks associated with the use of virtual server images from the public catalogs of a cloud service provider. In particular, we focus our investigation to the security problems of the public images available on the Amazon EC2 service. Over several months, we instantiated and analyzed over five thousands Linux and Windows images provided by the Amazon catalog, checking for a wide-range of security problems such as the prevalence of malware, the quantity of sensitive data left on such images, and the privacy risks of sharing an image on the cloud.

In particular, we identified three main threats related, respectively, to: 1) secure the image against external attacks, 2) secure the image against a malicious image provider, and 3) sanitize the image to prevent users from extracting and abusing private information left on the disk by the image provider. For example, in our experiments we identified many images in which a user can use standard tools to *undelete* files from the filesystem, and recover important documents including credentials and private keys.

Although public cloud server images are highly useful for organizations, if users are not properly trained, the risk associated with using these images can be quite high. The fact that these machines come pre-installed and pre-configured may communicate the wrong message, i.e., that they can provide an easy-to-use "shortcut" for users that do not have the skills to configure and setup a complex server. The reality is quite different, and this paper demonstrates that many different security considerations must be taken into account to make sure that a virtual image can be operated securely.

During our study we had continuous contact with the Amazon Web Services Security Team. Even though Amazon is not responsible of what users put into their images, the team has been prompt in addressing the security risks identified and described in this paper. Meanwhile, it has published public bulletins and tutorials to train users on how to use Amazon Machine Images (AMIs) in a secure way [22, 21]. A more detailed description of the Amazon feedback is provided in Section 6.

Finally, it is important to note that the security problems we describe in this work are general in nature, and they are not specific to the Amazon Cloud. We hope that our paper will raise awareness about the security risks of using and creating public software images on the cloud, and will encourage other cloud providers to verify and improve their security just as Amazon has done.

2 Overview of Amazon EC2

The Amazon *Elastic Compute Cloud (EC2)* is an Infrastructure-as-a-Service cloud provider where users can rent virtualized servers (called *instances*) on an hourly base. In particular, each user is allowed to run any pre-installed virtual machine image (called *Amazon Machine Image*, or *AMI*) on this service. To simplify the setup of a server, Amazon offers an online catalog where users can choose between a large number of AMIs that come pre-installed with common services such as web servers, web applications, and databases. An AMI can be created from a live system, a virtual machine image, or another AMI by copying the filesystem contents to the Amazon *Simple Storage Service (S3)* in a process called *bundling*. Public images may be available for free, or may be associated with a *product code* that allows companies to bill an additional usage cost via the Amazon *DevPay* payment service. Thus, some of these public machines are provided by companies, some are freely shared by single individuals, and some are created by the Amazon team itself.

In order to start an image, the user has to select a resource configuration (differing in processing, memory, and IO performance), a set of credentials that will be used for login, a firewall configuration for inbound connections (called a *security group*), and the *region* of the data center in which the machine will be started. Amazon data centers are currently located in the US (Northern Virginia and Northern California), Europe (Ireland), and Asia (Singapore). An additional data center (Tokyo) was added after we had completed our experiments. Hence, this data center will not be discussed in the rest of the paper.

When an AMI is instantiated, its public DNS address is announced via the Amazon API, and the machine is made accessible via SSH on port 22 (Linux) or Remote Desktop on port 3389 (Windows). An important aspect of this cloud computing service is that the instance's maintenance is completely under the responsibility of the user. That is, she is the one who can be held responsible for any content provided by the machine, and she is the one who has to assure its security. This includes, for example, the usual administration tasks of maintaining the configuration in a secure state (i.e., applying patches for vulnerable software, choosing the right passwords, and firewall configuration), and only allowing secure, encrypted communication protocols.

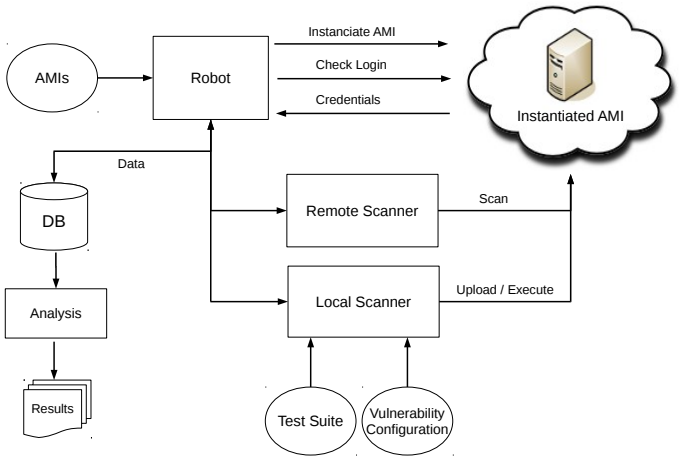

Figure 1: System Architecture

3 AMI Testing Methodology

To conduct our security evaluation, we developed an automated system to instantiate and test the Amazon's AMIs. The architecture of our system is highlighted in Fig. 1, and consists of three main components. The *Robot* is the part of the system that is responsible for instantiating the AMIs, and fetching the corresponding login credentials. Since Amazon does not control which credentials are configured in the public images, our tool was configured to try a list of the most common user names (e.g., `root`, `ec2-user`, `ubuntu`, and `bitnami` for Linux). Despite these attempts, there are cases in which the robot may fail to retrieve the correct login information. This is the case, for example, for AMIs whose credentials are distributed only to the image provider's customers by companies that make business by renting AMIs. Hence, these type of images are outside the scope of our evaluation.

After an AMI has been successfully instantiated by the robot, it is tested by two different scanners. The *Remote Scanner* collects the list of open ports[1] using the `NMap` tool [17], and downloads the index page of the installed web applications. In Section 5, we explain how an attacker can use this information as a fingerprint to identify running images. The *Local Scanner* component is responsible for uploading and running a set of tests. The test suite to be executed is packaged together in a self-extracting archive, uploaded to the AMI, and run on the machine with administrative privileges. In addition, the Local Scanner also analyzes the system for known vulnerabilities using the Nessus tool [23]. For AMIs running Microsoft Windows, the scripting of automated tasks is complicated by the limited remote adminis-

[1] Since Amazon does not allow external portscans of EC2 machines, we first established a virtual private network connection to the AMI through SSH, and then scanned the machine through this tunnel.

tration functionalities offered by the Windows environment. In this case, we mounted the remote disk and transfered the data using the `SMB/Netbios` subsystem. We then used the `psexec` tool [20] to execute remote commands and invoke the tests.

The test suite uploaded by the Local Scanner includes 24 tests grouped in 4 categories: general, network, privacy, and security.

The *general* category contains tests that collect general information about the system (e.g. the Linux distribution name, or the Windows version), the list of running processes, the file-system status (e.g., the mounted partitions), the list of installed packages, and the list of loaded kernel modules. In addition to these basic tests, the general category also contains scripts that save a copy of interesting data, such as emails (e.g., `/var/mail`), log files (e.g., `/var/log` and `%USER\Local Settings`), and installed web applications (e.g., `/var/www` and `HKEY_LOCAL_MACHINE\SOFTWARE`).

The *privacy* test cases focus on finding any sensitive information that may have been forgotten by the user that published the AMI. This includes, for example, unprotected private keys, application history files, shell history logs, and the content of the directory saved by the general test cases. Another important task of this test suite is to scan the filesystem to retrieve the contents of undeleted files.

The *network* test suite focuses on network-related information, such as shared directories and the list of open sockets. These lists, together with the processes bound to the sockets, can be used to verify if the image is establishing suspicious connections.

Finally, the *security* test suite consists of a number of well-known audit tools for Windows and Linux. Some of these tools look for the evidence of known rootkits, Trojans and backdoors (e.g. `Chkrootkit`, `RootkitHunter` and `RootkitRevealer`), while others specifically check for processes and sockets that have been hidden from the user (`PsTools/PsList` and `unhide`). In this phase, we also run the `ClamAV` antivirus software (see Section 4.2) to scan for the presence of known malware samples.

These security tests also contain checks for credentials that have been left or forgotten on the system (e.g., database passwords, login passwords, and SSH public keys). As already mentioned in an Amazon report published in June 2011 [11], these credentials could potentially be used as backdoors to allows attackers to log into running AMIs.

4 Results of the AMIs Analysis

Over a period of five months, between November 2010 to May 2011, we used our automated system to instantiate and analyze all Amazon images available in the Europe, Asia, US East, and US West data centers. In total, the catalog of these data centers contained 8,448 Linux AMIs and 1,202 Windows AMIs. Note that we were successfully able to analyze in depth a total of 5,303 AMIs. In the remaining cases, a number of technical problems prevented our tool to successfully complete the analysis. For example, sometimes an AMI did not start because the corresponding manifest file was missing, or corrupted. In some cases, the running image was not responding to SSH, or Remote Desktop connections. In other cases, the Amazon API failed to launch the machine, or our robot was not able to retrieve valid login credentials. These problems were particularly common for Windows machines where, in 45% of the images, the Amazon service was not able to provide us with a valid username and password to login into the machine. Nevertheless, we believe that a successful analysis of over 5,000 different images represents a sample large enough to be representative of the security and privacy status of publicly available AMIs.

In the rest of this section, we present and discuss the results of the individual test suites.

4.1 Software Vulnerabilities

The goal of this first phase of testing is to confirm the fact that the software running on each AMIs is often out of date and, therefore, must be immediately updated by the user after the image is instantiated.

For this purpose, we decided to run Nessus [23], an automated vulnerability scanner, on each AMI under test. In order to improve the accuracy of the results, our testing system provided Nessus with the image login credentials, so that the tool was able to perform a more precise local scan. In addition, to further reduce the false positives, the vulnerability scanner was automatically configured to run only the tests corresponding to the actual software installed on the machine. Nessus classifies each vulnerability with a *severity level* ranging from 0 to 3. Since we were not interested in analyzing each single vulnerability, but just in assessing the general security level of the software that was installed, we only considered vulnerabilities with the highest severity (e.g., *critical* vulnerabilities such as remote code execution).

From our analysis, 98% of Windows AMIs and 58% of Linux AMIs contain software with critical vulnerabilities. This observation was not typically restricted to a single application but often involved multiple services: an average of 46 for Windows and 11 for Linux images (the overall distribution is reported in Figure 2). On a broader scale, we observed that a large number of images come with software that is more than two years old. Our findings empirically demonstrate that renting and using an AMI without any adequate security assessment poses a real security risk for users. To further prove this point, in Section 4.2, we describe how one of the machines we were testing was probably compromised (مخترق) by an Internet malware in the short time that we were running our experiments.

We also looked at the most common vulnerabilities that affect Windows and Linux AMIs.

4.2 Security Risks

Malware

As part of our tests, we used ClamAV , an open source antivirus engine, to analyze the filesystem on the target AMI. ClamAV contains about 850,000 signatures to identify different types of known malware instances such as viruses, worms, spyware, and trojans. Since most of the existing malware targets the Windows operating systems, we analyzed the complete file-system tree of Windows AMIs, while we limited the coverage for Linux AMIs to common binary directories (e.g. `/usr/bin, /bin`, and `/sbin`). As a consequence, the scan time took on average of 40 minutes for a Windows installation, and less then a minute for a Linux one.

In our malware analysis, we discovered two infected AMIs, both Windows-based. The first machine was infected with a `Trojan-Spy` malware (variant 50112). This trojan has a wide range of capabilities, including performing key logging, monitoring processes on the computer, and stealing data

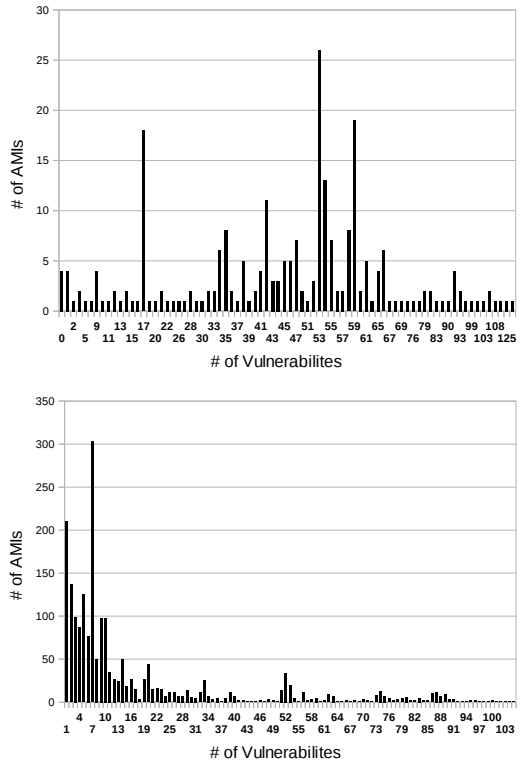

Figure 2: Distribution AMIs / Vulnerabilites (Windows and Linux)

from files saved on the machine. By manually analyzing this machine, we found that it was hosting different types of suspicious content such as `Trojan.Firepass`, a tool to decrypt and recover the passwords stored by Firefox. The second infected machine contained variant 173287 of the `Trojan.Agent` malware. This malware allows a malicious user to spy on the browsing habits of users, modify Internet Explorer settings, and download other malicious content.

While we were able to manually confirm the first case, we were unable to further analyze the second infected machine. In fact, after we rented it again for a manual analysis a few hours after the automated test, the infected files did not existed anymore. Hence, we believe that the AMI was most probably compromised by an automatically propagating malware during the time that we were executing our tests. In fact, the software vulnerability analysis showed that different services running on the machine suffered from known, remotely exploitable, vulnerabilities.

Unsolicited connections

Unsolicited outgoing connections from an invoked instance to an external address may be an indication for a significant security problem. For example, such connections could be the evidence of some kind of backdoor, or the sign for a malware infection. Outgoing connections that are more stealthy may also be used to gather information about the AMI's usage, and collect IP target addresses that can then be used to attack the instance through another built-in backdoor.

In our experiments, we observed several images that opened connections to various web applications within and outside of Amazon EC2. These connections were apparently checking for the availability of new versions of the installed software. Unfortunately, it is almost impossible to distinguish between a legitimate connection (e.g., a software update) and a connection that is used for malicious purposes.

Nevertheless, we noticed a number of suspicious connections on several Linux images: The Linux operating system comes with a service called *syslog* for recording various events generated by the system (e.g., the login and logout of users, the connection of hardware devices, or incoming requests toward the web server). Standard installations record these kinds of events in files usually stored under the `/var/log` directory and only users with administrative privileges are allowed to access the logs generated by the syslog service. In our tests, we discovered two AMIs in which the syslog daemon was configured to send the log messages to a remote host, out of the control of the user instantiating the image. It is clear that this setup constitutes a privacy breach, since confidential information, normally stored locally under a protected directory, were sent out to a third party machine.

Backdoors and Leftover Credentials

The primary mechanism to connect to a Linux machine remotely is through the `ssh` service. When a user rents an AMI, she is required to provide the public part of the her ssh key that it is then stored by Amazon in the `authorized_keys` in the home directory. The first problem with this process is that a user who is malicious and does not remove her public key from the image before making it public could login into any running instance of the AMI. The existence of these kinds of potential backdoors is known by Amazon since the beginning of April 2011 [19].

A second problem is related to the fact that the ssh server may also permit password-based authentication, thus providing a similar backdoor functionality if the AMI provider does not remove her passwords from the machine. In addition, while leftover ssh keys only allow people with the corresponding private key (normally the AMI image creator), to obtain access to the instance, passwords provide a larger attack vector: Anybody can extract the password hashes from an AMI, and try to crack them using a password-cracking tool (e.g., John the Ripper [9]).

In other words, ssh keys were probably left on the images by mistake, and without a malicious intent. The same applies to password, with the difference that passwords can also be exploited by third parties, transforming a mistake in a serious security problem.

During our tests, we gathered these leftover credentials, and performed an analysis to verify if a remote login would be possible by checking the account information in `/etc/passwd` and `/etc/shadow`, as well as the remote access configuration of OpenSSH.

The results, summarized in Table 1, show that the problem of leftover credentials is significant: 21.8% of the scanned AMIs contain leftover credentials that would allow a third-party to remotely login into the machine. The table also reports the type of credentials, and lists how many of these would grant superuser privileges (either via root, sudo or su with a password).

4.3 Privacy Risks

The sharing of AMIs not only bears risks for the customers who rent them, but also for the user who creates and dis-

	East	West	EU	Asia	Total
AMIs (%)	34.8	8.4	9.8	6.3	21.8
With Passwd	67	10	22	2	101
With SSH keys	794	53	86	32	965
With Both	71	6	9	4	90
Superuser Priv.	783	57	105	26	971
User Priv.	149	12	12	12	185

Table 1: Left credentials per AMI

Finding	Total	Image	Remote
Amazon RDS	4	0	4
dDNS	1	0	1
SQL	7	6	1
MySql	58	45	13
WebApp	3	2	1
VNC	1	1	0
Total	74	54	20

Table 2: Credentials in history files

tributes the image. In fact, if the image contains sensitive information, this would be available to anybody who is renting the AMI. For example, an attacker can gather SSH private keys to break into other machines, or use forgotten Amazon Web Services (AWS) keys to start instances at the image provider's cost. In addition, other data sources such as the browser and shell histories, or the database of last login attempts can be used to identify and de-anonymize the AMI's creator.

Private keys

We developed a number of tests to search the AMIs' filesystem for typical filenames used to store keys (e.g., `id_dsa` and `id_rsa` for SSH keys, and `pk-[0-9A-Z]*.pem` and `cert-[0-9A-Z]*.pem` for AWS API keys). Our system was able to identify 67 Amazon API keys, and 56 private SSH keys that were forgotten. The API keys are not password protected and, therefore, can immediately be used to start images on the cloud at the expense of the key's owner. Even though it is good security practice to protect SSH keys with a passphrase, 54 out of 56 keys were not protected. Thus, these keys are easily reusable by anybody who has access to them. Although some of the keys may have been generated specifically to install and configure the AMI, it would not be a surprising discovery if some users reused their own personal key, or use the key on the AMI to access other hosts, or Amazon images.

By consulting the last login attempts (i.e., by `lastlog` or `last` commands), an attacker can easily retrieve IP addresses that likely belong to other machines owned by the same person. Our analysis showed that 22% of the analyzed AMIs contain information in at least one of the `last` login databases. The `lastb` database contains the failed login attempts, and therefore, can also be very helpful in retrieving user account passwords since passwords that are mistyped or typed too early often appear as user names in this database. There were 187 AMIs that contained a total of 66,601 entries in their `lastb` databases. Note that host names gathered from the shell history, the SSH user configuration, and the SSH server connection logs can also provide useful clues to an attacker.

Browser History

Nine AMIs contained a Firefox history file (two concerning root and seven concerning a normal user). Note that because of ethical concerns, we did not manually inspect the contents of the browser history. Rather, we used scripts to check which domains had been contacted. From the automated analysis of the history file, we discovered that one machine was used by a person to log into the portal of a Fortune 500 company. The same user then logged into his/her personal Google email account. Combining this kind of information, history files can easily be used to de-anonymize, and reveal information about the image's creator.

Shell History

When we tested the AMI using our test suite, we inspected common shell history files (e.g. `~/.history`, `~/.bash_history`, `~/.sh_history`) that were left on the image when it was created. We discovered that 612 AMIs (i.e., 11.54% of the total) contained at least one single history file. We found a total of 869 files that stored interesting information (471 for root and 398 for generic users), and that contained 158,354 lines of command history. In these logs, we identified 74 different authentication credentials that were specified in the command line, and consequently recorded on file (ref. Table 2).

For example, the standard MySQL client allows to specify the password from the command line using the `-p` flag. A similar scenario occurs when sensitive information, such as a password or a credit card number, is transferred to a web application using an HTTP GET request. GET requests, contrary to POST submissions, are stored on the web server's logs. The credentials we discovered belong to two categories: local and remote.

The credentials in the *image* group grant an attacker access to a service/resource that is hosted on the AMI. In contrast, *remote* credentials enable the access to a remote target. For example, we identified remote credentials that can be used to modify (and access) the domain name information of a dynamic DNS account. A malicious user that obtains a DNS management password can easily change the DNS configuration, and redirect the traffic of the original host to his own machines. In addition, we discovered four credentials for the Amazon Relational Database Service (RDS) – a web service to set up, operate, and scale a relational database in the Amazon cloud. We also found credentials for local and remote web applications for different uses (e.g. Evergreen, GlassFish, and Vertica) and for a database performance monitoring service. One machine was configured with VNC, and its password was specified from the command line. Finally, we were able to collect 13 credentials for MySQL that were used in the authentication of remote databases.

Recovery of deleted files

In the previous sections, we discussed the types of sensitive information that may be forgotten by the image provider. Unfortunately, the simple solution of deleting this information before making the image publicly available is not satisfactory from a security point of view.

In many file systems, when a user deletes a file, the space occupied by the file is marked as free, but the content of the file physically remains on the media (e.g. the hard-disk). The contents of the deleted file are definitely lost only when

this marked space is overwritten by another file. Utilities such as `shred`, `wipe`, `sfill`, `scrub` and `zerofree` make data recovery difficult either by overwriting the file's contents before the file is actually unlinked, or by overwriting all the corresponding empty blocks in the filesystem (i.e., secure deletion or wiping). When these security mechanisms are not used, it is possible to use tools (e.g., `extundelete` and `Winundelete`) to attempt to recover previously deleted files.

In the context of Amazon EC2, in order to publish a custom image on the Amazon Cloud, a user has to prepare her image using a predefined procedure called *bundling*. This procedure involves three main steps: Create an image from a loopback device or a mounted filesystem, compress and encrypt the image, and finally, split it into manageable parts so that it can be uploaded to the S3 storage.

The first step of this procedure changes across different bundling methods adopted by the user (ref. Table 3). For example, the `ec2-bundle-image` method is used to bundle an image that was prepared in a loopback file. In this case, the tool transfers the data to the image using a block level operation (e.g. similar to the `dd` utility). In contrast, if the user wishes to bundle a running system, she can choose the `ec2-bundle-vol` tool that creates the image by recursively copying files from the live filesystem (e.g., using `rsync`). In this case, the bundle system works at the file level.

Any filesystem image created with a block-level tool will also contain blocks marked as free, and thus may contain parts of deleted files. As a result, out of the four bundling methods provided by Amazon, three are prone to a file undeletion attack.

To show that our concerns have practical security implications, we randomly selected 1,100 Linux AMIs in four different regions (US East/West, Europe and Asia). We then used the `extundelete` data recovery utility to analyze the filesystem, and recover the contents of all deleted files. In our experiment, we were able to recover files for 98% of the AMIs (from a minimum of 6 to a maximum of more than 40,000 files per AMI). In total, we were able to retrieve 28.3GB of data (i.e., an average of 24MB per AMI).

We collected statistics on the type (Table 4) of the undeleted files by remotely running the `file` command. Note that in order to protect the privacy of Amazon users, we did not access the contents of the recovered data, and we also did not transfer this data out of the vulnerable AMI. The table shows a breakdown of the types of sensitive data we were able to retrieve (e.g., PDFs, Office documents, private keys). Again, note that the Amazon AWS keys are not password-protected. That is, an attacker that gains access to these keys is then able to instantiate Amazon resources (e.g. S3 and AWS services) at the victim's expense (i.e., the costs are charged to the victim's credit card).

In our analysis, we verified if the same problem exists for Windows AMIs. We analyzed some images using the `WinUndelete` tool , and were able to recover deleted files in all cases. Interestingly, we were also able to undeleted 8,996 files from an official image that was published by Amazon AWS itself.

5 Machine Fingerprinting

In the previous sections, we presented a number of experiments we conducted to assess the security and privacy issues involved in the release and use of public AMIs. The results of our experiments showed that a large number of factors must be considered when making sure that a virtual machine image can be operated securely (e.g., services must be patched and information must be sanitized).

Method	Level	Vulnerable
`ec2-bundle-vol`	File-System	No
`ec2-bundle-image`	Block	Yes
From AMI snapshot	Block	Yes
From VMWare	Block	Yes

Table 3: Tested Bundle Methods

Type	#
Home files (`/home, /root`)	33,011
Images (min. 800x600)	1,085
Microsoft Office documents	336
Amazon AWS certificates and access keys	293
SSH private keys	232
PGP/GPG private keys	151
PDF documents	141
Password file (`/etc/shadow`)	106

Table 4: Recovered data from deleted files

A number of the issues we described in the previous sections could potentially be exploited by an attacker (or a malicious image provider) to obtain unauthorized remote access to any running machine that adopted a certain vulnerable AMI. However, finding the right target is not necessarily an easy task.

In order to explore the feasibility, from an attacker point of view, of automatically matching a running instance back to the corresponding AMI, we started our experiment by querying different public IP registries (ARIN, RIPE, and LAPNIC) to obtained a list of all IPs belonging to the Amazon EC2 service for the regions US East/West, Europe and Asia. The result was a set of sub-networks that comprises 653,401 distinct IPs that are potentially associated with running images.

For each IP, we queried the status of thirty commonly used ports and compared the results with the information extracted from the AMI analysis. We only queried a limited number of ports because our aim was to be as non-intrusive as possible. For the same reason, we configured NMap to only send a few packets per second to prevent any flooding, or denial of service effect.

Our scan detected 233,228 running instances. This number may not reflect the exact number of instances there were indeed running. That is, there may have been virtual machines that might have been blocking all ports.

We adopted three different approaches to match and map a running instance to a set of possible AMIs. The three methods are based on the comparison of the SSH keys, versions of network services, and web-application signatures.

Table 5 depicts the results obtained by applying the three techniques. The first column shows the number of running instances to which a certain technique could be applied (e.g., the number of instances where we were able to grab the SSH banner). The last two columns report the number of running machines for which a certain matching approach was able to reduce the set of candidate AMIs to either 10 or 50 per matched instance.

SSH matching Every SSH server has a host key that is used to identify itself. The public part of this key is used

Approach	Instances	Candidates 1	10	50
SSH	130,580	2,149	8,869	11,762
Services	203,563	7,017	30,345	63,512
Web	125,554	5,548	31,651	54,918

Table 5: Discovered Instances

to verify the authenticity of the server. Therefore, this key is disclosed to the clients. In the EC2, the host key of an image needs to be regenerated upon instantiation of an AMI for two reasons: First, a host key that is shared among several machines makes these servers vulnerable to man-in-the-middle attacks (i.e., especially when the private host key is freely accessible). Second, an unaltered host key can serve as an identifier for the AMI, and may thus convey sensitive information about the software that is used in the instance.

This key regeneration operation is normally performed by the `cloud-init` script provided by Amazon. The script should normally be invoked at startup when the image is first booted. However, if the image provider either forgets or intentionally decides not to add the script to his AMI, this important initialization procedure is not performed. In such cases, it is very easy for an attacker to match the SSH keys extracted from the AMIs with the ones obtained from a simple `NMap` scan. As reported in Table 5, we were able to precisely identify over 2,100 AMI instances by using this method.

Service matching In the cases where the ssh-based identification failed, we attempted to compare the banners captured by `NMap` with the information extracted from the services installed on the AMIs. In particular, we compared the service name, the service version, and (optionally) the additional information fields returned by the thirty common ports we scanned in our experiment.

The service-matching approach is not as precise as the ssh-based identification. Hence, it may produce false positives if the user has modified the running services. However, since most services installed on the AMIs were old and out of date, it is very unlikely that new services (or updated ones) will match the same banners as the one extracted from the AMIs. Therefore, a service update will likely decrease the matching rate, but unlikely generate false positives. The fact that over 7,000 machines were identified using this method seems to support the hypothesis that a large number of users often forget to update the installed software after they rent an AMI.

Web matching For our last AMI matching approach, we first collected web information from all the instances that had ports 80 and 443 (i.e., web ports) open. We then compared this information with the data we collected during the scan of the Amazon AMIs.

In the first phase, we used the `WhatWeb` tool to extract the name and version of the installed web server, the configuration details (e.g., the OpenSSL version), and the installed interpreters (e.g., PHP, and JSP). In addition, we also attempted to detect the name and version of the web applications installed in the root document of the web server by using WhatWeb's plugins for the detection of over 900 popular web software.

In the second phase, we compared this information to the scanned AMIs, and checked for those machines that had the same web server, the same configuration, and the same versions of the language interpreters. Since different installations of the same operating system distribution likely share this information, we then further reduced the size of the candidate set by checking the web application name detected by the WhatWeb tool.

The last row of Table 5 shows that we were able to identify more than 5,000 machines by using this technique.

6 Amazon's Feedback

Clearly, one question that arises is if it is ethically acceptable and justifiable to conduct experiments on a real cloud service. During all our experiments, we took into account the privacy of the users, the sensitivity of the data that was analyzed, and the availability of Amazon's services. In addition, all our AMI tests were conducted by automated tools running inside virtual machines we rented explicitly for our study.

Amazon has a dedicated group dealing with the security issues of their cloud computing infrastructure: the AWS (Amazon Web Services) Security Team. We first contacted them on May 19th 2011, and provided information about the credentials that were inadvertently left on public AMIs. Amazon immediately verified and acknowledged the problem, and contacted all the affected customers as summarized by a public bulletin released on June 4th [22]. In cases where the affected customer could not be reached immediately, the security team acted on behalf of the user, and changed the status of the vulnerable AMI to private to prevent further exposure of the customer's personal credentials. We also communicated to the AWS Security team our concerns regarding the privacy issues related to publishing of public AMIs (e.g., history files, remote logging, and left-over private keys). The security team reacted quickly, and realeased a tutorial [21] within five days to help customers share public images in a secure manner. Finally, we contacted again Amazon on June 24th about the possibility of recovering deleted data from the public Amazon AMIs. To fix the problem, we provided them some of the countermeasures we discussed in Section 4.3. Their team immediately reported the issue internally and was grateful of the issue we reported to attention. By the time of writing, Amazon has already verified all the public AMIs where we have been able to recover data, and has moved on to check the status of all other public AMIs. The AWS security team is also working on providing a solution to prevent the recovery of private documents by undeletion.

The second part of our experiments included the use of a port scanner to scan running images. Even though port scanning has not been considered to be illegal per se (e.g., such as in the legal ruling in [1]), this activity may be considered an ethically sensitive issue. However, given the limited number of ports scanned (i.e, 30) and the very low volume of packets per second that we generated, we believe that our activity could not have caused any damage to the integrity and availability of Amazon's network, or the images running on it.

7 Related Work

There are several organizations that released general security guidance on the usage of cloud computing, such as [2, 7]. Amazon Web Services, in addition to the security bulletins

already mentioned, released a paper describing the security processes put in place, focusing more specifically on the management of public images [3]. The problem statement of security on cloud computing infrastructures has been widely explored. Garfinkel and Rosenblum [14] studied the problem statement of using virtual images and especially the security problems of using third party virtual images. Glott and al. [15] present a good overview of the problem of sharing images as well. The above papers expose a set of best practices and problem statements but did not perform any practical assessment tests.

Another approach to address the problem consists in specifying contracts between cloud providers and virtual machine users [18]. In this case, the authors describe extensions to the Open Virtual Machine Format (OVF) to specify the requirements of machine images in order to be safely executed by a cloud computing provider. Wei et al [24] propose a technique to share and use third party images in a secure way. scanning.

More specific to Amazon EC2, a novel approach was proposed by Bleikerts et al. [12]. The paper analyses the security of an infrastructure (a set of connected virtual machines) deployed on Amazon EC2 through graph theory techniques. The goal is to provide security resilience metrics at the infrastructure level rather than at the virtual image level. Thus, this study aims at designing better cloud infrastructures by identifying problems at the configuration of the security groups (network firewall rules). Slaviero et al. [16] showed that cloud users are not careful when choosing AMIs. By publishing a public malicious AMI, they showed that this AMI was instantiated several times and statistics about the usage of the AMIs were collected. Moreover, they showed that it was possible to circumvent the payment mechanism of paid AMIs by modifying the AMI manifest file.

Finally, concurrently and in parallel to our work, Bugiel et al. [13] have recently conducted a study in which they perform similar experiments on the Amazon's EC2 catalogue and have reached similar conclusions. Note, however, that our experiments are more comprehensive and have been conducted on a larger scale. While they have only considered 1255 AMIs, we selected and automatically analyzed over 5000 public images provided by Amazon in four distinct data centers. We also discovered and discussed a wider number of security issues by testing every image for known malware samples and vulnerabilities. Furthermore, we collaborated closely with Amazon's Security Team to have the identified problems acknowledged and fixed.

8 Acknowledgments

This research has been partially funded by the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n. 257007, by the NSF (CNS-1116777) and by COMET K1, FFG - Austrian Research Promotion Agency and the Secure Business Austria. The authors would like to acknowledge Don Bailey and the Amazon Web Services Security Team for their commitment in addressing the security risks we identified.

9 References

[1] Security focus: Port scans legal, judge says. http://www.securityfocus.com/news/126.

[2] Security guidance for critical areas of focus in cloud computing v2.1, Dec. 2009. https://cloudsecurityalliance.org/csaguide.pdf.

[3] Amazon elastic compute cloud, May 2011. http://aws.amazon.com/security.

[4] Amazon elastic compute cloud (amazon ec2), May 2011. http://aws.amazon.com/ec2/.

[5] Cloud computing, cloud hosting and online storage by rackspace hosting, May 2011. http://www.rackspace.com/cloud/.

[6] Enterprise cloud computing from terremark, May 2011. http://www.terremark.com/services/cloudcomputing.aspx.

[7] Guidelines on security and privacy in public cloud computing, draft special publication 800-144, 2011.

[8] Ibm smartcloud, May 2011. http://www.ibm.com/cloud-computing/us/en/#!iaas.

[9] John the ripper unix password cracker, Apr. 2011. http://www.openwall.com/john/.

[10] Joyent smartdatacenter, May 2011. http://www.joyent.com/services/smartdatacenter-services/.

[11] Reminder about safely sharing and using public amis, Jun 2011. http://aws.amazon.com/security/security-bulletins/reminder-about-safely-sharing-and-using-public-amis/.

[12] S. Bleikertz, M. Schunter, C. Probst, D. Pendarakis, and K. Eriksson. Security audits of multi-tier virtual infrastructures in public infrastructure clouds. In *Proceedings of the 2010 ACM workshop on Cloud computing security workshop*.

[13] S. Bugiel, S. Nürnberger, T. Pöppelmann, A. Sadeghi, and T. Schneider. Amazonia: When elasticity snaps back. 2011.

[14] T. Garfinkel and M. Rosenblum. When virtual is harder than real: Security challenges in virtual machine based computing environments. In *Proceedings of the 10th conference on Hot Topics in Operating Systems-Volume 10*.

[15] R. Glott, E. Husmann, A. Sadeghi, and M. Schunter. Trustworthy clouds underpinning the future internet. *The Future Internet*.

[16] M. S. Haroon Meer, Nick Arvanitis. Clobbering the cloud, part 4 of 5, 2009. http://www.sensepost.com/labs/conferences/clobbering_the_cloud/amazon.

[17] I. LLC. Network Mapper (NMap), 1996-2011. http://nmap.org/.

[18] J. Matthews, T. Garfinkel, C. Hoff, and J. Wheeler. Virtual machine contracts for datacenter and cloud computing environments. In *Proceedings of the 1st workshop on Automated control for datacenters and clouds*.

[19] A. Puzic. Cloud security: Amazon's ec2 serves up 'certified pre-owned' server images, Apr. 2011. http://dvlabs.tippingpoint.com/blog/2011/04/11/cloud-security-amazons-ec2-serves-up-certified-pre-owned

[20] M. Russinovich. Psexec, 2009. http://technet.microsoft.com/en-us/sysinternals/bb897553.

[21] A. A. Security. How to share and use public amis in a secure manner, June 2011. http://aws.amazon.com/articles/0155828273219400.

[22] A. A. Security. Reminder about safely sharing and using public amis, June 2011. http://aws.amazon.com/security/security-bulletins/reminder-about-safely-sharing-and-using-public-amis/.

[23] T. N. Security. Nessus vulnerability scanner, 2002-2011. http://www.tenable.com/products/nessus.

[24] J. Wei, X. Zhang, G. Ammons, V. Bala, and P. Ning. Managing security of virtual machine images in a cloud environment. In *Proceedings of the 2009 ACM workshop on Cloud computing security*.