Security, Privacy and Cloud Computing

Jose Tomas Robles Hahn
Supervisor: Ralph Holz
Future Internet Seminar - Winter Term 2010/2011
Chair for Network Architectures and Services
Faculty of Computer Science, Technische Universität München
Email: jtrh@mytum.de

ABSTRACT
Cloud computing usage is growing every day as users discover its many advantages. However, the move to the cloud exposes users to new security and privacy threats. We give details on cloud-specific vulnerabilities and caveats and show some technical ways to address them. We also note that users cannot rely on technology alone to completely solve all their problems: how to protect against government surveillance remains an open question.

Keywords
Cloud computing, Security, Privacy, Surveillance, Encryption

1. INTRODUCTION
In the 1980s, the emergence of the personal computer (PC) revolutionized the relationship between society and computers. Before the PC, computers were the domain of governments, large enterprises and academic institutions. The massification of the personal computer, together with the Internet, enabled a new world of possibilities: electronic commerce, 24 hour access to information located anywhere on Earth, and much more.

Today, we are again experiencing an exciting paradigm shift in the IT industry. Cloud computing solutions are being massively adopted by governments, businesses and consumers. The market research company IDC forecasts a threefold growth of customer spending on cloud computing services, reaching US$ 42 billion by 2012 [1].

Cloud computing brings many benefits to its users. Businesses and government agencies can enjoy substantial cost-savings in IT infrastructure and increased flexibility to react to changes in requirements of computing power. Consumers are provided with ubiquitous access to their data and redundant storage, which protects them against inconveniences such as having to carry their computer everywhere, or data loss caused by a failing, non-backed up disk drive [2].

Unfortunately, certain groups are exploiting cloud computing to the detriment of its legitimate users. We are talking about hackers and governments: These two groups both wish to covertly access users' data, but for different reasons. Computer criminals want to access data such as credit card numbers, bank login credentials, financial records and other confidential information in order to gain profit, while governments want access to that data in the name of fighting crime and terrorism. Cloud computing puts users' data at a higher risk of being accessed by unauthorized individuals, since their information is now stored in datacenters which they do not control. This also allows governments to invade users' privacy by getting their data directly from the cloud provider, without informing the affected user [2].

In this article we will discuss the security and privacy risks that users face by moving their data to the cloud and show how we can use technology to solve them. We will base our discussion on Christopher Soghoian's article *Caught in the Cloud: Privacy, Encryption, and Government Back Doors in the Web 2.0 Era* [2].

This article is organized as follows: First, in chapter 2 we will talk about the definition and characteristics of cloud computing. We will also talk about the benefits that cloud computing brings to the table. Then, in chapter 3 we will explore the security challenges cloud computing faces, the technologies that exist to solve those challenges, and some actual security attacks. Finally, we discuss cloud computing privacy issues in chapter 4.

2. CLOUD COMPUTING
2.1 Defining cloud computing
In the last decades, the computing paradigm has evolved substantially. Voas and Zhang identified six distinct phases [3]. The first phase corresponds to the traditional terminal era, where mainframes shared by many users did all the hard computing work. In phase two came the now ubiquitous personal computer. In the next phase, local networks appeared. Phase four came to be with the interconnection of local networks, which ultimately resulted in the Internet. In the following phase came distributed computing. Finally, in phase six, cloud computing made its appearance, opening the door to access a potentially unlimited amount of computing resources, in a scalable and simple way.

But, what is cloud computing? How do we define it? Unfortunately, so many different products in the market are associated to cloud computing by their respective manufacturers, that it becomes difficult to specify what cloud computing actually is. Larry Ellison, co-founder and CEO of Oracle Corporation, when asked about cloud computing while at a conference, said *"I have no idea what anyone is talking about"*, *"We've redefined cloud computing to include everything that we already do"*, and *"I can't think of anything that isn't cloud computing with all of these announcements"* [4].

doi: 10.2313/NET-2011-05-2_05

A notable definition was created by the U.S. National Institute of Standards and Technology (NIST) [5]. According to their definition, cloud computing has five essential characteristics: on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service. They also define three cloud service models (which can be viewed as layers, as shown in figure 1), which must be deployed on top of cloud infrastructure that has the five essential characteristics mentioned above:

- Cloud Software as a Service (SaaS): Provider offers users access to its application over a network. Usually, this is implemented as a Web application. Providers in this category include Facebook [6], Google Apps [7] and Google Mail [8].
- Cloud Platform as a Service (PaaS): Provider offers a platform where users can deploy their own applications. Some well-known providers in this category are Microsoft Windows Azure Platform [9] and Google App Engine [10].
- Cloud Infrastructure as a Service (IaaS): Provider offers computing resources such as processing power, storage and network capacity. Some providers in this category include the Amazon Elastic Compute Cloud (Amazon EC2) [11] and Rackspace Cloud Servers [12].

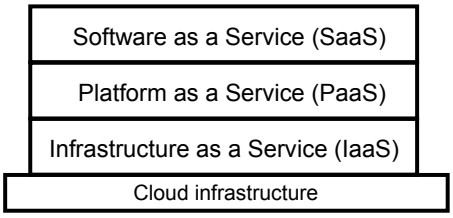

Figure 1: Cloud computing service models viewed as layers, deployed on top of cloud infrastructure that has NIST's five essential cloud computing characteristics. (Image source: Own work)

As in Soghoian's article [2], this paper will focus on Web applications executed in a Web browser, where the application's code is downloaded as needed from a remote server that also stores users' files. So keep in mind that when we talk about cloud computing applications, we will be referring to Web applications.

2.2 Cloud computing in practice

In the personal computing paradigm, users' can run locally installed applications like word processors, spreadsheets, personal financial management software, picture organizers, and so on. Thus, their data is always stored locally, i.e. in their own computer [2]. They maintain physical control over their data and therefore must assume the responsibility which that entails: For example, they have to take measures themselves to keep their data safe from hardware failures (e.g. making backups regularly), and they have to ensure that their data is available at any place they need it (e.g. copying it to a USB flash drive).

Today, users are increasingly moving their data to the cloud. Email, which requires Internet access to check for new messages, was not surprisingly, the first application to move [2]. Some time later, other applications became available online, such as office suites like Google Apps [7] and Microsoft Office Web Apps [13], and picture editors, like Adobe Photoshop Express [14].

We will now discuss the benefits of cloud computing for service providers, businesses and consumers.

- Cloud service providers benefit from cloud computing because it solves the software piracy problem, since part of the software code resides exclusively on the providers' servers. For the same reason, trade secrets such as in-house developed algorithms are also safe from reverse engineering [2].

  A purely economic advantage is vendor lock-in: as customers cannot easily take their data from one provider to a competitor (unless the provider itself provides them with data export functionality, and the competitor with import functionality), they are discouraged from changing cloud providers [15].

  In the case of paid services, another economic advantage is the subscription payment model, where customers make periodic payments to the provider, which in the long term can amount to much higher revenues than one-time purchases typical of classic software products [16].
- Business users benefit from not having to maintain a datacenter, redundant storage, having less IT personnel costs and turning IT infrastructure from a fixed cost into a variable cost. Another important advantage of cloud computing is the flexibility to acquire additional computing capacity only as required, saving money on investments made to support infrequently occurring peak loads.
- Consumers benefit from many cloud service providers offering their Web applications for "free" or cheaper than their desktop counterparts. Note that in this case, free often means subjecting oneself to targeted advertising and data mining [17]; and that cheaper does not necessarily mean long term savings [16] (after some time, the customer may have already paid the cost of a one-time software purchase). Another benefit is that the cloud provider handles backups and hardware failures. A useful feature in today's mobile world is having ubiquitous access to your data: anywhere in the world, just an Internet connection is needed, even a mobile phone is enough.

As Internet is not available everywhere yet, offline access to data stored in the cloud is an important feature that Web applications do not normally have. To solve this issue, Google created Gears [18], which allows Web applications like Google Mail to store a copy of users' mail on their computers. Google announced in 2010 [19] that Gears would be replaced in the future by HTML5 Web Storage [20].

As Web applications look more and more like their desktop counterparts, it is becoming increasingly difficult for users

doi: 10.2313/NET-2011-05-2_05

to realize whether they are using a desktop application or a Web application, and where their data is actually stored [2]. This confusion highlights the importance of security and privacy in cloud computing: If users are going to use Web applications without noticing it, then cloud computing should be made as secure and private as possible.

3. CLOUD COMPUTING SECURITY

Security is an important subject in today's networked world, and cloud computing, still in its infancy, is no exception. Two years ago, a survey by IDC [21] showed that security was the top challenge ascribed to the cloud computing model. Thus, security vulnerabilities can even affect the decision of whether to adopt a cloud computing solution at all.

In this chapter we will discuss the concept of cloud-specific vulnerabilities, followed by which issues are affecting cloud security together with security technologies to solve them, and finally, some concrete security attacks.

3.1 Cloud-specific vulnerabilities

Cloud Web applications depend on widely used technologies, such as DNS, TLS and Web browsers. This means that the vulnerabilities of those technologies could *also* be considered vulnerabilities of Web applications. But, are there any vulnerabilities *specific* to cloud computing? How can we classify a vulnerability as *cloud-specific*?

Grobauer, Walloschek and Stocker attempted to answer that question in [22]. They consider a vulnerability to be cloud-specific if at least one of the following conditions is met:

- The vulnerability is intrinsic to or prevalent in a core cloud technology, such as Web applications and services, virtualization and cryptography.
- The vulnerability's root cause is one of the five essential cloud characteristics identified by NIST (see also section 2.1 and [5]).
- The vulnerability is caused by cloud innovations making the implementation of security best practices difficult.
- The vulnerability is common in most modern cloud offerings.

Although the following security weaknesses and vulnerabilities also apply to non-cloud technologies, we conclude from the above conditions that they can be considered cloud-specific.

3.2 Security weaknesses in cloud computing

In this section we will talk about security weaknesses and caveats affecting cloud-related technologies.

3.2.1 Cloud providers fail to provide encryption to their users

Soghoian has strongly criticized cloud service providers for not providing encrypted access to their Web applications [2]. For example, webmail providers such as Yahoo! Mail [23], encrypt their login page with HTTPS, but then revert to plain HTTP. Although users' passwords remain protected, session cookies, together with the reading and writing of email messages, are transmitted *in the clear* (i.e. unencrypted). Another example is Facebook, which encrypts the transmission of login credentials, but not the login page itself. Therefore, Facebook users cannot easily verify that they are filling the form on the real Facebook website.

Soghoian argues that there is no economic incentive for cloud providers to provide encrypted access and encrypted storage by default [2]. One reason for this is the higher operating cost of encrypted access, which demands more processor time per client connection to sustain the same number of unencrypted connections, requiring additional hardware purchases in order to keep quality of service constant. If the cloud service is provided for "free", then there is even less incentive for cloud providers to provide encryption, as providing free service is not free for them. To pay for their costs, free cloud providers mine users' data so they can show highly targeted advertisements to users [17]. If that data were stored encrypted, it would not be possible to analyze it for advertising purposes.

Our last point concerns market demand: If users do not demand encryption from cloud providers, they will probably never offer it. A reason for this situation is lack of information: Cloud providers do not openly disclose to users the risks to which their data is subject [2]. From insider attacks [42] to government surveillance, there are enough reasons to desire encryption (for more information, see also chapter 4).

3.2.2 Man-in-the-middle attacks

This is an attack form in which the attacker redirects traffic between a client and a server through him, so that he can log and possibly also alter the communication. Both client and server believe they are talking directly to each other. This attack is normally implemented by tricking the client into connecting to the attacker instead of the desired server and then relaying the traffic to the real destination [24]. Man-in-the-middle (MITM) attacks can be perpetrated by forging DNS packets, DNS cache poisoning, or ARP spoofing, for example. DNSSEC and HTTPS/TLS are two technologies that can prevent MITM attacks (see sections 3.3.4 and 3.3.2, respectively).

3.2.3 Data encryption caveats

Before implementing a data encryption technology, some important questions need to be considered. Their possible answers illustrate the limits of data encryption.

- Where will the encryption key be stored? If the cloud provider is in possession of the key, the customer must trust the service provider not to use it for unauthorized purposes and to store it safely outside the reach of hackers. Furthermore, it is important to know that the cloud service provider may be forced by law enforcement to disclose the encryption key to them, sometimes without being allowed to inform the customer (using a so-called *gag order*) [25]. An example of a

doi: 10.2313/NET-2011-05-2_05

cloud service provider that stores customer data in encrypted form is Hushmail [26].

- Where will the encryption and decryption processes be performed? If the cloud service provider is storing the key, then it will also perform the encryption and decryption. Further discussion deserves the case where the customer is in sole possession of her key. On the one hand, if encryption and decryption procedures are executed exclusively on the customer's premises, then it is guaranteed that the cloud provider does not have access to the encrypted data even for a single instant. On the other hand, if the customer supplies her key to the cloud provider each time she needs to encrypt or decrypt data, so that the cloud provider performs the encryption or decryption and then deletes the customer's key from memory, then the customer's key is at risk.

  An example that shows the importance of this question is the case of a drug dealer that used Hushmail's secure email service [27]. Hushmail offers both a server-side and a client-side encryption mode. The more secure client-side encryption mode is done using an open source Java applet, while the less secure server-side encryption mode is performed through a webmail interface where the client supplies the passphrase to his key when needed, which is immediately deleted from memory after use. Law enforcement officials ordered Hushmail to record the customer's passphrase instead of deleting it from memory and then used it to decrypt all the customer's mail. Note that the open source Java applet would not have saved the drug dealer, because law enforcement can also order Hushmail to supply the customer with a backdoored applet. To be safe from that backdoor, a client would have to read and compile the applet's source himself, which is more work than just accepting Hushmail's compiled binary.

Irrespective of which choices are made regarding the above two questions, we also want to make you aware of the following caveat, which also applies to network encryption: encryption is not a magic solution; encrypted data can be stored indefinitely until enough computing power is available to decrypt it. What today is considered impossible may be feasible in, say, five or ten years.

3.2.4 User interface attacks

Web applications are accessed through a Web browser, so the browser's user interface becomes an important security factor.

One kind of user interface attack is that in which an attacker tries to fool the user into thinking that she is visiting a real website instead of a forgery. Techniques used here include fake HTTPS lock icons, which are only detected by attentive users [38], homographic attacks with international characters that look like certain national characters [38], and browser software vulnerabilities, which can trick the browser into showing incorrect information, like a fake URL in the address bar.

In section 3.4.2 we will look at a security attack that exploits the Web browser's user interface.

3.3 Security measures in cloud computing

We will now present some security technologies that are relevant to cloud applications.

3.3.1 Single site browsers

Users do not need to download or install Web applications — they just execute them in a Web browser. That same Web browser is also used to interact with sensitive websites such as banks or webmail. Browsers also store a history of all the websites a user visited, and often, website passwords. All that information, stored in a single place, is at risk of being stolen by hackers exploiting Web browser vulnerabilities [2].

Single site browsers, also known as site-specific browsers, seek to reduce that risk by creating a separate browser instance for a Web application. The most advanced single site browser technology is Mozilla Prism for Firefox [28], which allows users to create a dedicated shortcut on their desktop for a Web application, which will open a dedicated browser window. This dedicated browser instance maintains its own preferences and user data, which is safe against access by malicious websites and Web applications running in separate Prism sessions or Firefox windows.

However, single site browsers are not all about security. They also improve usability by hiding user interface elements such as the toolbar and the address bar [2]. This makes sense, since the back and forward buttons are document-oriented and therefore not suitable for an application. Even though these usability improvements lower the adoption barrier of Web applications, they come at a price. Since all user interface elements are hidden, users have no way to verify that they are connected through a secure connection [2]. It remains to be seen how Web browser vendors attempt to solve this problem.

3.3.2 Network encryption: HTTPS, TLS and PKI

Network encryption protects data as it travels from one computer to another. Once data arrives at its destination, it may either be stored encrypted (see section 3.3.3) or unencrypted.

The most popular network encryption technology used to secure communication between cloud clients and Web applications is the Hypertext Transfer Protocol Secure (HTTPS), which is a combination of HTTP with the Transport Layer Security (TLS) [29] protocol. TLS resulted from the standardization of the Secure Sockets Layer 3.0 protocol. It provides confidentiality, integrity and authentication between clients and servers.

The most common authentication method used on the Web today is the Web server providing the Web browser with a certificate, which contains its public key together with a digital signature that binds it with an identity. Since the Web server's certificate is provided through an insecure channel, the Web browser must have a way to verify that the certificate it received actually came from the Web server it is talking to. This is assured through a Public Key Infrastructure (PKI). In a PKI, an independent, trusted entity called a Certificate Authority (CA) is in charge of verifying that a certain public key is associated to a certain identity. This attestation is provided in form of a digital signature with

doi: 10.2313/NET-2011-05-2_05

the CA's private key. Web browsers verify this digital signature using the CA's own certificate, which is supposed to be obtained through a secure channel, but normally comes pre-installed together with the operating system or browser.

TLS certificates may be revoked for various reasons. A revocation means that the CA does not consider the certificate to be valid anymore. Possible reasons may include, for example, fraudulently obtained certificates or a legitimate certificate owner's private key being compromised. TLS implementations, such as Web browsers, need some way to verify that a certificate has not been revoked. One method to accomplish this task are Certificate Revocation Lists (CRLs) [30], which are issued periodically by CAs and contain a signed timestamped list of serial numbers of revoked certificates. A disadvantage of CRLs is that revocation reports will not be published until the next periodic update. For critical applications, such a delay may not be acceptable. The Online Certificate Status Protocol (OCSP) [31] is an alternative to CRLs that makes revocations available as soon as they are issued by the CA. All modern Web browsers try to check the certificates they receive with OCSP before accepting them.

3.3.3 Data storage encryption

Cloud service providers store users' data on servers outside the control of their customers. Customer data is at risk of being accessed by unauthorized individuals such as hackers, thieves, and even datacenter employees [42]. Encryption is the tool of choice to protect that data.

Some important aspects to consider before implementing data storage encryption were discussed in section 3.2.3.

3.3.4 DNS security: DNSSEC

When a user types a Web application's URL in her Web browser's address bar (e.g. `http://www.facebook.com/`), one of the first actions the Web browser takes is using DNS to find out which IP address (`69.63.190.18`) corresponds to the hostname in the URL (`www.facebook.com`).

DNS can be viewed as a tree-structured distributed hierarchical database of *zones* [32]. A zone is an independent administrative entity that contains resource records (RRs) describing many different types of information, like IP address-to-name mappings and delegations. Delegation RRs indicate that a zone (the *parent zone*) has assigned responsibility for a certain subset of it (the *child zone*) to a different name server.

Now let us examine the process of finding out the mapping of a name to an IP address: First, the Web browser (a *DNS client*) contacts a stub resolver provided by the operating system, which in turn contacts a local recursive or resolving name server (a *resolver*). The resolver then queries successively all necessary name servers in the hierarchy starting from the root zone and stopping either at the requested RR or at an error. Finally, the result is passed to the stub resolver, which then informs the browser about either the requested IP address or a look up error. Of course these steps were greatly simplified, for example, by ignoring DNS caches present on client computers and resolvers.

We now know why DNS security is important for cloud computing: the mapping of names to IP addresses. If a malicious attacker could somehow manage to modify the IP address that a client received when trying to access `www.facebook.com`, he could redirect to his own server all packets sent by the client and then forward them to Facebook's server, of course after recording or altering them. This is the man-in-the-middle attack we mentioned in section 3.2.2. We will now present a DNS extension designed to ensure that DNS answers cannot be altered in transit.

The DNS Security Extensions (DNSSEC) [33] extend DNS by adding authentication and integrity to DNS RRs through a hierarchy of cryptographic digital signatures [32]. Resolvers can verify the digital signatures attached to DNS RRs they receive by following a chain of trust of public keys and digital signatures that starts at the root zone and goes through zone delegations until finally reaching the name server that stores the RR to be authenticated (see figure 2).

To achieve its task, the DNSSEC specification defines four new RR types [34]:

- DNSKEY (DNS Public Key): This RR stores the public key corresponding to the private key used to sign the zone's resource record sets (RRsets). Resolvers use the public key to validate and authenticate those RRsets. For the zones illustrated in figure 2, there are DNSKEY RRs in the root, `se.` and `nic.se.` zones.

- DS (Delegation Signer): The DS RR is stored in a parent zone and points to the DNSKEY RR of the name server responsible for the child zone. That DNSKEY RR is stored only in the child zone's name server. Because the DS RR is signed, a chain of trust consisting of linked DS and DNSKEY RRs is formed. For the zones illustrated in figure 2, there are two DS RRs: one in the root zone (pointing to the DNSKEY RR of the `se.` zone) and one in `se.` (pointing to the DNSKEY RR of `nic.se.`).

- RRSIG (Resource Record Digital Signature): RRSIG RRs store digital signatures for RRsets, which can be of any RR type, including the other three DNSSEC RR types and mappings of names to IP addresses. In figure 2, there would be RRSIG RRs signing the DNSKEY and DS RRs of `.`, `se.` and `nic.se.`, and the RR mapping `www.nic.se.` to `212.247.7.218`.

- NSEC (Next Secure): This RR type is used to authenticate *not found* answers. NSEC RRs form a chain linking all RRs in a zone, so that if a resolver requests a non-existent RR, it receives as answer an NSEC RR containing information about the two chained RRs that come *before* and *after* the requested RR (see also *Canonical Form and Order of Resource Records* in [34]). Since no RRs can exist between any two RRs of an NSEC chain, a resolver can be sure that the requested RR does not exist. Unfortunately this NSEC chain also allows attackers to follow the entire chain and obtain a list of all RRs in a zone. To solve this privacy and security problem, hashed names are used in the new NSEC3 RR [35].

doi: 10.2313/NET-2011-05-2_05

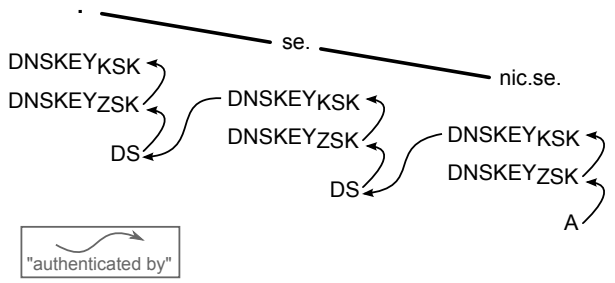

Figure 2: Following the DNSSEC chain of trust — An example: An *A* resource record (RR) mapping the name *www.nic.se.* to the IP address *212.247.7.218* is authenticated by the zone signing key (ZSK) of the *nic.se.* zone, which in turn is authenticated by the key signing key (KSK) of the same zone. The *DS* RR in zone *se.* authenticates the KSK of zone *nic.se.* and is authenticated by the ZSK of zone *se.*, which in turn is authenticated by the KSK of the same zone. In the root zone, there is a *DS* RR authenticating the KSK of the *se.* top-level domain (TLD). This *DS* RR is authenticated by the root zone's ZSK, which in turn is authenticated by the KSK of the same zone. Because the chain ends at the root zone, the authentication of the root zone's KSK must be performed by other means. (Image source: Own work)

We note that while DNSSEC provides end-to-end authentication (answers remain authenticated as they pass through intermediate DNS servers and are therefore protected against man-in-the-middle attacks), no hop-to-hop encryption is provided (an attacker is able to read requests and answers while they are in transit). DNSCurve, which encrypts packets but does not digitally sign them, complements DNSSEC by providing privacy for DNS traffic [36].

We end our discussion of DNSSEC with an interesting idea proposed in an RFC: Since DNSSEC can protect any RR type, we could distribute general-purpose certificates stored in signed CERT RRs, which could be used for applications such as secure email. This would provide an alternative to classic PKIs. More about this can be found in [37].

3.4 Security attacks

In this section we will present four security attacks that are of particular relevance to Web applications. They are concrete examples that show how the security weaknesses we discussed in section can be exploited in practice.

3.4.1 Signing TLS certificates with another site's certificate

TLS site certificates are supposed to be signed only by certificate authorities (CAs) or intermediate CAs designated by them. In this attack, Marlinspike discovered that it was possible to sign a TLS site certificate for *any* website of the attacker's choice, with just a site certificate legitimately obtained from an established certificate authority (CA), and have most browsers and other TLS implementations accept it as valid even though a site's certificate is not supposed to be able to sign other certificates [38]. This vulnerability was possible because even though TLS certificates can have their `BasicConstraints` field set to `CA:FALSE` (which means that the certificate cannot be used to sign other certificates), most CAs either did not bother to include that field at all or did not declare it as `critical` (fields set as critical must be obeyed by TLS implementations). Another reason was that most Web browsers and TLS implementations did not bother to check that field, even if it was present.

Marlinspike created a tool called *sslsniff* to exploit this vulnerability in an automated fashion. *Sslsniff*, supplied with a legitimately obtained site certificate for any domain, carries out a man-in-the-middle attack, intercepting HTTPS traffic and generating a certificate for the destination website signed with the supplied legitimate site certificate on the fly. Even though this vulnerability has been corrected in the meantime, *sslsniff* can still be useful as a general man-in-the-middle tool for TLS.

3.4.2 HTTPS via HTTP attack

The security researcher Moxie Marlinspike made the observation that most people arrive at HTTPS sites after being sent from an HTTP site [38]. Specifically, he named two ways: Clicking on links (or submitting a form) and through HTTP 302 redirects. For example, many online banking login pages are transmitted through HTTP. These pages normally either contain an IFRAME with an HTTPS login form or a form that posts the login credentials to an HTTPS URL. Facebook's login form [6] also features an insecure form that posts to an HTTPS URL.

Marlinspike proposed to attack not the HTTPS connection, as would be usual, but to attack the *HTTP* connection. For this purpose he created *sslstrip*, which, like *sslsniff*, is a man-in-the-middle attack tool. *Sslstrip* watches HTTP traffic looking for `https://...` links and changes them to `http://...` links pointing to a Web server controlled by the attacker, keeping track of what was changed. The attacker's Web server proxies the HTTP requests as HTTPS to the destination Web server and also rewrites all `https://...` links sent back to the client. Web browsers do not show any warning because only HTTP traffic is seen by them. For additional believability, *sslstrip* can watch out for favicon requests and send a lock icon to make the fake website look even more real.

3.4.3 Null prefix TLS attack [39] [40]

In a TLS certificate, the website hostname to which it belongs is specified in the Common Name (CN) field in the subject of the certificate.

Certificate Authorities (CAs), before signing a certificate, normally verify ownership of the domain name specified in the CN field, without caring about any subdomains and without verifying other subject information like Organization (legal name of the subject) or Country.

The CN field is represented as a Pascal string, which in its memory representation specifies first the length of the string, and then the string itself. This is different from C strings, which are just sequences of characters that are terminated by a single null character. The structure of Pascal strings

doi: 10.2313/NET-2011-05-2_05

has the effect that null characters are treated as any other character. Marlinspike observed this characteristic and realized that he could include null characters in the CN field, so that for example, he could generate a certificate signing request (CSR) for `www.facebook.com\0.attacker.org`. A certificate authority will ignore the null character in the CN and only verify the ownership of the `attacker.org` domain, because, as we said above, the CA does not care about subdomains. This verification procedure is usually just an email message to the registered contact of the domain `attacker.org`, which of course the attacker himself controls.

Of course this attack is not yet complete as we have not said anything about the role of the Web browser. Marlinspike noticed that most TLS implementations treat the CN field as a C string, using C string comparison and manipulation functions. This means that when the browser compares the CN specified in the certificate (`www.facebook.com\0.attacker.org`) with the current website hostname (`www.facebook.com`), the comparison functions stops at the null character and returns *equal*.

If the certificate authority were to revoke the certificate for `www.facebook.com\0.attacker.org`, we would need to use Marlinspike's OCSP attack, which we will explain in the following section.

3.4.4 OCSP attack

The Online Certificate Status Protocol (OCSP) [31] enables Web browsers and other TLS implementations to verify that a legitimately obtained certificate is still considered valid by the certificate authority.

When a Web browser receives an apparently valid certificate from a Web server, before accepting it, it sends a verification request to the OCSP server specified in the certificate. The OCSP server sends a response that includes a response status (which can be *successful*, *malformedRequest*, *internalError*, *tryLater*, *sigRequired* or *unauthorized*) and a signed response data field.

Marlinspike noticed that even though a fake *successful* response status would fail due to the signature in the response field, the innocent-looking response status *tryLater* does not require a signature and can therefore be faked without difficulties [39] [41]. Most Web browsers, after receiving a *tryLater* response status code, give the certificate the benefit of the doubt, accepting it without alerting the user.

4. CLOUD COMPUTING PRIVACY

Having your personal data stored in a place outside your control is becoming commonplace thanks to cloud computing. In this final chapter we will briefly discuss the privacy challenges that cloud computing faces, together with the relationship between government and the cloud. We will also show how technology can help users regain control of their privacy.

4.1 Privacy challenges in the cloud

In this section we will briefly cover some aspects of cloud computing that may affect users' privacy.

A fundamental characteristic of cloud computing is having users' private data outside their physical control. This can have many consequences. For example, data could be mined by the cloud provider with, or even worse, without authorization. Excessively curious cloud datacenter employees could read users' private (unencrypted) data without their knowledge. A recent case involving Google demonstrates that these privacy risks must be taken seriously: A (now ex-)Google employee was caught spying on teen users, accessing their Google Voice call logs, chat transcripts and contact lists [42].

Another aspect to consider when storing data in a cloud provider datacenter is of a legal nature: Since users' data may be stored in a datacenter located anywhere in the world, their data could be stored in a foreign country without the user ever noticing. A foreign government could, for example, use surveillance techniques to help their companies gain unauthorized access to trade secrets. Leaving covert surveillance aside, users' must take into account that the laws of the government where the datacenter is located may be different from the laws of the country where the user lives.

The challenges presented in this section could be solved through network *and* data encryption [2]. However, in section 4.2 we will introduce a different kind of adversary, one which cannot be defeated solely through encryption.

4.2 Government and the cloud

We will now focus on privacy threats coming not from private actors, such as hackers, but from the government.

4.2.1 Situation in the United States

In the United States the government has been continuously expanding its use of surveillance. Soghoian argues that this is happening because technology has drastically lowered the cost of spying on its citizens [2]. He proposes encryption as the definite solution against government intrusion, but recognizes that the government can force a cloud provider to insert backdoors into its software in order to circumvent the encryption.

One of the most important legal tools used by the U.S. Government to force cloud providers to hand them users' private data is the third-party doctrine. Other relevant laws include the Wiretap Act, the All Writs Act and the Foreign Intelligence Surveillance Act [2].

The Fourth Amendment to the United States Constitution protects U.S. citizens against unreasonable search and seizure, dependent upon a person's reasonable expectation of privacy. Unfortunately, the Fourth Amendment does not protect data stored in the cloud. The third-party doctrine establishes that a person does not have an expectation of privacy regarding information they share with others [2]. Courts consider that a user giving data to a cloud provider is sharing the data with them.

Until now we have not shown an actual example of what information the government can obtain from a cloud provider. Facebook's *Subpoena / Search Warrant Guidelines* [43] and Microsoft's *Global Criminal Compliance Handbook* [44] offer a few glimpses of that. For example, Facebook can supply

doi: 10.2313/NET-2011-05-2_05

law enforcement with a user's complete profile information and uploaded photos, irrespective of her privacy settings.

4.2.2 Situation in Germany

The United States is not alone regarding intrusions into their citizen's privacy. Even though Germany fares very well in Forrester Research's Data Protection Heat Map [45], that map bases its evaluation on each country's data protection laws, which do not cover government surveillance. For a taste of what the German government can do or wants to do, read the following:

§§111 and 112 of the 2004 Telecommunications Act (*Telekommunikationsgesetz* in German) [46] allow the government to force telecommunication service providers (which include cloud service providers like webmail) to hand over information such as a customer's name, address, birthdate, and email address, without a court order, through an automated query system that includes a search function in case law enforcement has incomplete request data. The admissibility of such a query is a decision of the requesting law enforcement authority.

New surveillance laws, such as the Federal Criminal Office Law (*BKA-Gesetz* in German) are explained in layman's terms on *Freedom instead of Fear*'s website (*Freiheit statt Angst* in German) [47]. This law gives the Federal Criminal Office new powers that are usually available only to state police and secret services.

An actual example of court-ordered surveillance in Germany is the Java Anonymous Proxy (JAP), which is an open source software for anonymously browsing websites. A court order obtained by the German Federal Office of Criminal Investigation ordered the JAP developers to add a backdoor to log accesses to an illegal website [2]. In this specific case, the open source nature of JAP allowed a user to discover the backdoor, showing that open source software cannot be modified to insert a backdoor without some technically advanced user noticing it in the source code [2].

4.2.3 Compelled certificate creation attack

This attack, explained in detail in Soghoian and Stamm's article *Certified Lies: Detecting and Defeating Government Interception Attacks Against SSL* [49], is about government agencies forcing certificate authorities (CAs) to issue false TLS certificates to enable intelligence agencies to covertly watch an individual's secure Web communications. They state that all popular browsers and operating systems come pre-installed with more than 100 CA root certificates from entities all over the world, including government-owned CAs. They created a Firefox add-on called *CertLock* that caches certificates from sites that the user visits in order to detect suspicious changes, focusing on changes of the country of the CA that issued the certificate. However, the Firefox add-on has still not been released. An add-on with a more extensive approach than *CertLock* is *Certificate Patrol* [50], which warns the user every time a site sends a certificate different from the cached copy.

4.2.4 Compelled backdoor attack

As we said in section 4.2.1, government can force a cloud provider to insert a backdoor into its application, in order to bypass any encryption the service may provide.

One difficulty users face is that, unlike desktop applications that have a version number and do not automatically update themselves, cloud applications can change anytime without the user noticing [2]. A possible solution for this is Web application fingerprinting. Through the external analysis of the Web application, it may be possible to generate a fingerprint identifying a unique Web application and version pair. Kozina, Golub and Gros developed a fingerprinting method in [51], which compares link patterns, forms and keywords. However, the proposed method has currently some important limitations that prevent it from being ready for prime time.

5. CONCLUSION

It is clear that cloud computing is here to stay [1].

There are many challenges that we can only face if we understand what we are dealing with, how it may affect us and which possible solutions exist. In this article we have covered those points:

We found a concrete definition of the cloud computing concept, which is necessary if we ought to study it. We saw the advantages and disadvantages of the cloud for different actors: cloud service providers, businesses and individual consumers. We covered various security technologies that can solve many cloud computing security challenges. But we must convince cloud providers and users of the importance of implementing available security technologies. A small success in that aspect was achieved when Google was convinced to enable HTTPS by default in its Google Mail service. Unfortunately, the absence of economic incentives for providers to implement effective security measures means that this challenge has yet to be solved. The threat of certain security attacks serves to remind us that security technologies are not perfect.

We have also learned that privacy is more relevant than ever for users of cloud services, and deserves as much attention as security. Users must learn which privacy threats from hackers, data miners and government exist, so that they can take an informed decision when moving their private data to the cloud.

As cloud computing is still in its infancy, much remains yet to be seen. If we inform ourselves of the challenges and solutions that we face, we will be able to tackle them successfully.

6. REFERENCES

[1] *IT Cloud Services Forecast — 2008, 2012: A Key Driver of New Growth*, International Data Corporation, USA, 2008.
Available at `http://blogs.idc.com/ie/?p=224`

[2] C. Soghoian: *Caught in the Cloud: Privacy, Encryption, and Government Back Doors in the Web 2.0 Era*, In Journal on Telecommunications and High Technology Law, vol. 8, no. 2, Boulder, Colorado, USA, 2010.
Available at
`http://www.jthtl.org/articles.php?volume=8`

doi: 10.2313/NET-2011-05-2_05

[3] J. Voas, J. Zhang: *Cloud Computing: New Wine or Just a New Bottle?*, In IT Professional, vol. 11, no. 2, pp. 15–17, March–April, USA, 2009

[4] G. Fowler and B. Worthen: *The internet industry is on a cloud — whatever that may mean*, In Wall Street Journal, March 26, USA, 2009.
Available at `http://online.wsj.com/article/SB123802623665542725.html`

[5] P. Mell, T. Grance: *Effectively and Securely Using the Cloud Computing Paradigm*, version 26, National Institute of Standards and Technology (NIST), USA, 2009.
Available at `http://csrc.nist.gov/groups/SNS/cloud-computing/`

[6] *Facebook*, `http://www.facebook.com/`

[7] *Google Apps*, `http://www.google.com/apps/`

[8] *Google Mail*, `https://mail.google.com/`

[9] *Microsoft Windows Azure Platform*, `http://www.microsoft.com/windowsazure/`

[10] *Google App Engine*, `http://appengine.google.com/`

[11] *Amazon Elastic Compute Cloud (Amazon EC2)*, `http://aws.amazon.com/ec2/`

[12] *Rackspace Cloud*, `http://www.rackspacecloud.com/`

[13] *Microsoft Office Web Apps*, `http://office.microsoft.com/web-apps/`

[14] *Adobe Photoshop Express*, `http://www.photoshop.com/tools`

[15] M. Brandel: *Cloud computing: Don't get caught without an exit strategy*, In Computerworld, March 3, USA, 2009.
Available at `http://www.computerworld.com/s/article/9128665/`

[16] J. D. Lashar: *The Hidden Cost of SaaS*, In destinationCRM.com, May 1, USA, 2008.
Available at `http://www.destinationcrm.com/Articles/Columns-Departments/The-Tipping-Point/The-Hidden-Cost-of-SaaS---48682.aspx`

[17] *About Google Mail: More on Google Mail and privacy*, `https://mail.google.com/mail/help/about_privacy.html`

[18] *Google Gears*, `http://gears.google.com/`

[19] I. Fette: *Hello HTML5*, In Gears API Blog, February 19, USA, 2010.
Available at `http://gearsblog.blogspot.com/2010/02/hello-html5.html`

[20] *HTML5 Web Storage*, `http://www.w3.org/TR/webstorage/`

[21] *IDC Enterprise Panel*, n=244, August, 2008

[22] B. Grobauer, T. Walloschek, E. Stocker: *Understanding Cloud-Computing Vulnerabilities*, In Security & Privacy, IEEE, vol. PP, no. 99, p. 1, 2010

[23] *Yahoo! Mail*, `http://mail.yahoo.com/`

[24] E. Cole: *Network Security Bible 2nd Edition*, p. 130, Wiley, USA, 2009.

[25] Anonymous author: *My National Security Letter Gag Order*, In The Washington Post, March 23, USA, 2007.
Available at `http://www.washingtonpost.com/wp-dyn/content/article/2007/03/22/AR2007032201882.html`

[26] *How Hushmail Can Protect You*, `http://www.hushmail.com/about/technology/security/`

[27] R. Singel: *Encrypted E-Mail Company Hushmail Spills to Feds*, In Wired News — Threat Level, November 7, USA, 2007.
Available at `http://www.wired.com/threatlevel/2007/11/encrypted-e-mai/`

[28] *Mozilla Prism*, `http://prism.mozilla.com/`

[29] T. Dierks, E. Rescorla: *The Transport Layer Security (TLS) Protocol Version 1.2*, IETF RFC 5246, 2008.
Available at `http://tools.ietf.org/html/rfc5246`

[30] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, W. Polk: *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, IETF RFC 5280, 2008.
Available at `http://tools.ietf.org/html/rfc5280`

[31] M. Myers, R. Ankney, A. Malpani, S. Galperin, C. Adams: *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*, IETF RFC 2560, 1999.
Available at `http://tools.ietf.org/html/rfc2560`

[32] A. Friedlander, A. Mankin, W. D. Maughan, S. D. Crocker: *DNSSEC: a protocol toward securing the internet infrastructure*, In Commun. ACM 50, 6 (June), 44–50, 2007.
Available at `http://doi.acm.org/10.1145/1247001.1247004`

[33] R. Arends, R. Austein, M. Larson, D. Massey, S. Rose: *DNS Security Introduction and Requirements*, IETF RFC 4033, 2005.
Available at `http://tools.ietf.org/html/rfc4033`

[34] R. Arends, R. Austein, M. Larson, D. Massey, S. Rose: *Resource Records for the DNS Security Extensions*, IETF RFC 4034, 2005.
Available at `http://tools.ietf.org/html/rfc4034`

[35] B. Laurie, G. Sisson, R. Arends, D. Blacka: *DNS Security (DNSSEC) Hashed Authenticated Denial of Existence*, IETF RFC 5155, 2008.
Available at `http://tools.ietf.org/html/rfc5155`

[36] W. C. A. Wijngaards, B. J. Overeinder: *Securing DNS: Extending DNS Servers with a DNSSEC Validator*, In Security & Privacy, IEEE, vol. 7, no. 5, pp. 36–43, Sept.–Oct., 2009

[37] S. Josefsson: *Storing Certificates in the Domain Name System (DNS)*, IETF RFC 4398, 2006.
Available at `http://tools.ietf.org/html/rfc4398`

[38] M. Marlinspike: *New Tricks for Defeating SSL in Practice*, In Black Hat DC 2009, February, Washington DC, USA, 2009.
Available at `https://www.blackhat.com/presentations/bh-dc-09/Marlinspike/BlackHat-DC-09-Marlinspike-Defeating-SSL.pdf`

[39] M. Marlinspike: *More Tricks for Defeating SSL in Practice*, In Black Hat USA 2009, July, Las Vegas, Nevada, USA, 2009.
Available at
https://www.blackhat.com/presentations/bh-usa-09/MARLINSPIKE/BHUSA09-Marlinspike-DefeatSSL-SLIDES.pdf

[40] M. Marlinspike: *Null Prefix Attacks Against SSL/TLS Certificates*, July 29, 2009.
Available at
http://www.thoughtcrime.org/papers/null-prefix-attacks.pdf

[41] M. Marlinspike: *Defeating OCSP With The Character '3'*, July 29, 2009.
Available at
http://www.thoughtcrime.org/papers/ocsp-attack.pdf

[42] K. Zetter: *Ex-Googler Allegedly Spied on User E-Mails, Chats*, In Wired News — Threat Level, September 15, USA, 2010.
Available at
http://www.wired.com/threatlevel/2010/09/google-spy/

[43] *Facebook Subpoena / Search Warrant Guidelines*,
http://cryptome.org/isp-spy/facebook-spy.pdf

[44] *Microsoft Online Services — Global Criminal Compliance Handbook (U.S. Domestic Version)*,
http://cryptome.org/isp-spy/microsoft-spy.zip

[45] *Interactive Data Protection Heat Map*,
http://www.forrester.com/cloudprivacyheatmap

[46] *Telekommunikationsgesetz vom 22. Juni 2004 (BGBl. I S. 1190)*, zuletzt geändert durch Artikel 2 des Gesetzes vom 17. Februar 2010 (BGBl. I S. 78).
Available at
http://www.gesetze-im-internet.de/tkg_2004/

[47] *BKA-Gesetz — Aktion Freiheit statt Angst e.V.*,
http://www.aktion-freiheitstattangst.org/de/themen/polizei-geheimdienste-a-militaer/97-bka-gesetz

[48] T. C. Greene: *Net anonymity service back-doored*, In The Register, August 21, UK, 2003.
Available at
http://www.theregister.co.uk/2003/08/21/net_anonymity_service_backdoored/

[49] C. Soghoian, S. Stamm: *Certified Lies: Detecting and Defeating Government Interception Attacks Against SSL*, Under Submission.
Available at http://www.dubfire.net/#pubs

[50] *Certificate Patrol — a psyced Firefox/Mozilla add-on*,
http://patrol.psyced.org/

[51] M. Kozina, M. Golub, S. Gros: *A method for identifying Web applications*, In International Journal of Information Security, vol. 8, no. 6, pp. 455–467, Springer Berlin/Heidelberg, 2009. DOI: 10.1007/s10207-009-0092-3
Available at http://www.springerlink.com/content/wu204731658576t4/