On the feasibility of automated semantic attacks in the cloud

Ryan Heartfield and George Loukas

Abstract While existing security mechanisms often work well against most known attack types, they are typically incapable of addressing semantic attacks. Such attacks bypass technical protection systems by exploiting the emotional response of the users in unusual technical configurations rather than by focussing on specific technical vulnerabilities. We show that semantic attacks can easily be performed in a cloud environment, where applications that would traditionally be run locally may now require interaction with an online system shared by several users. We illustrate the feasibility of an automated semantic attack in a popular cloud storage environment, evaluate its impact and provide recommendations for defending against such attacks.

1 Introduction

Cyber criminals are often adept at manipulating people into giving them access to information they need [1]. In an information security context, one can achieve deception by exploiting stereotypical thinking, processing ability, inexperience, truth bias and other semantic attack vectors [7]. We explore the applicability of such a semantic attack in cloud computing, with a proof of concept prototype worm which utilises variants of semantic attack vectors to propagate from one system to another. Our work illustrates the relative simplicity of automating such an attack effectively and without considerable complexity, time or expertise. Our particular case study is on the increasingly popular service of cloud-based storage.

Ryan Heartfield
University of Greenwich, e-mail: hr811@greenwich.ac.uk

George Loukas
University of Greenwich, e-mail: g.loukas@greenwich.ac.uk

2 Related Work

Chen and Katz have recently argued that several of the usual underlying information security issues and challenges, such as phishing, downtime, password weaknesses and compromised hosts, not only remain but are often amplified in a cloud environment [9]. Cloud storage services provide applications that allow users and organisations to store information in local directories, synchronised and backed up in the cloud, available to access via web browsers or by installing particular applications on other machines. Chen and Katz have highlighted that this type of shared resource environment constitutes a security issue that is specific to cloud computing [9]. Mulazzani et al. recently showed how such cloud shared storage can be used as an attack platform, identifying in particular an exploitation of the popular Dropbox service [4]. During installation, Dropbox uses a unique host ID to authenticate a device to a user's account. The ID can be stolen via a social engineering guise, such as a spoofed email with a link to a rogue website. This compromises the Dropbox account and gives the attacker full access to all its content. The service itself is not directly attacked, but becomes the deception platform of the attack. Our aim is to illustrate that such social engineering attacks are not only applicable in the cloud, but can even be automated if combined with a worm and a complementary deception infrastructure, without considerable expertise or effort.

3 Worm-based semantic attack

Our implementation assumes that the targets are the users of a popular cloud-based storage service in its usual default set-up in a Microsoft Windows operating system, but makes no assumption as to the technical competency of these users. We have chosen DropBox and SugarSync as the cloud services for our case-study due exclusively to their popularity and not any technical vulnerability, as we target the human rather than the technical aspects of the system. Such services provide simple software applications and web access portals that synchronise local folders to the cloud storage, essentially storing the files on a dedicated storage location that is available from anywhere, at any time, backed up and protected with up-to-date technical security systems. DropBox has an estimated 10 million users and SugarSync is used in almost every country in the world. An automated semantic attack using both would provide a wealth of prospective victims. To achieve such an attack, we developed a novel worm and a complementary deception infrastructure, including spoofed/phishing websites and scareware (Figure 1).

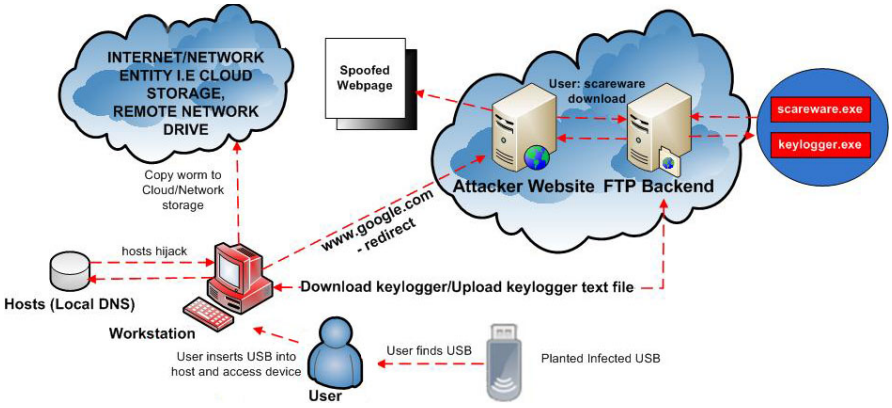

Fig. 1 Attack anatomy

3.1 The test-bed

Due to obvious ethical considerations, the case study worm was contained during development and experimental testing in a sandbox virtual network environment, physically disconnected from the Internet. For this purpose, we implemented the Hyper-V sandbox hypervisor test environment within Microsoft Windows Server 2008 R2. This enables the provisioning of virtual machines within a contained virtual network, without access to external resources outside of the virtual abstraction layer within Hyper-V. This internal network allows communication between virtual machines only and no communication between virtual machines and the hypervisor host or external entities. Internal network communication is performed through a virtual switch in the abstraction layer which transports data between synthetic virtual network interfaces assigned to each virtual machine, essentially producing a workgroup network environment with files replicated and accessible by several users on multiple systems (Figure 2).

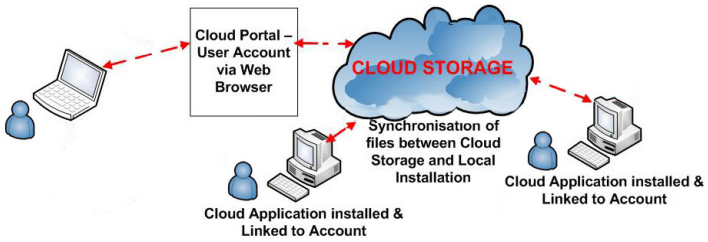

Fig. 2 Architecture of the Cloud environment

3.2 The worm's function and payload

The case study worm relies on manipulation of truth bias [8], essentially deceiving the user to assume a file to be another legitimate one, using a file masquerading technique. The original file is deleted and the malformed file takes its place. Figure 3 shows the finite state machine diagram for the worm.

By seeing the expected filename and extension, the user is likely to believe at first impression that this is the original file, especially as there is no other file with similar name and the user themselves have not removed the file. Should the user notice that the file has changed in any way, such as its icon or description, they are now likely to be curious as to why this has happened when the filename appears correct. Both these emotive reactions (truth bias and curiosity) entice the user to open the file and by doing so execute the worm program. At this point the worm will run without visual execution or further interaction with the user, who may now believe that the file is corrupted. The worm has hidden within the user's profile directory and on next login will execute the rest of its payload, including more file masquerading deceptions, a keylogger etc.

The attack may also involve a USB media semantic attack vector, with the worm running as a hidden executable autorun file. The semantic vector here is curiosity and desire to explore the USB memory device. Current trends in worms demonstrate operational relationships with hierarchical overlay systems (Zeus, Storm, Conficker), where the worm can be defined as an agent

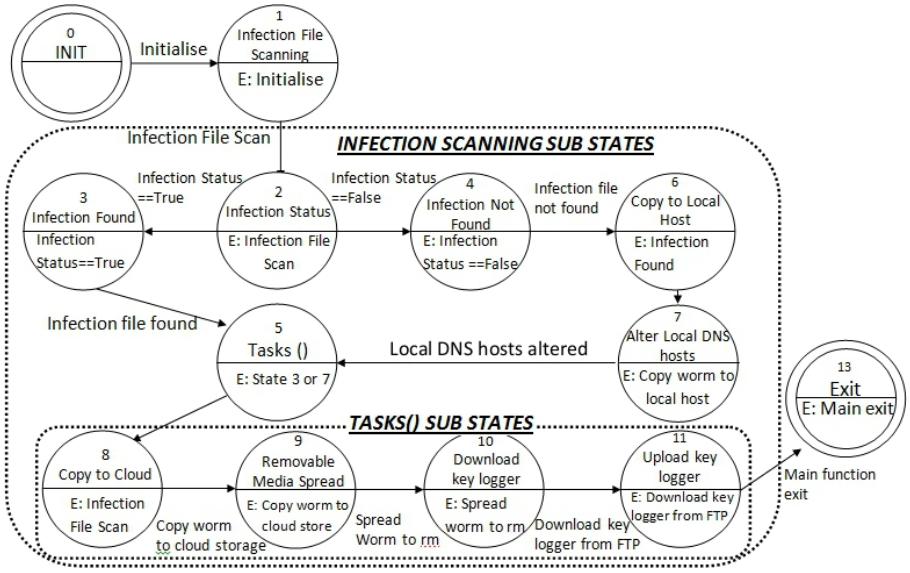

Fig. 3 The worm's finite state machine

of a parent system to deliver or direct a specific attack. To depict this, we introduce a typical malicious backend architecture consisting of a spoofed website (Google Search Engine) and scareware application (fictitious Google anti-virus), which illustrates how the worm can be used for fraud, data theft and other exploitations.

4 Experimental evaluation

Before evaluating the impact of our automated semantic attack we first tested the deception element. 20 students, randomly selected from our university's computing programmes for a fictitious survey, were given a brief overview of the use of Cloud storage and were asked to create a number of files of different types within the Dropbox/SugarSync folder, containing personal data. The system was reset and the user was asked to locate each file and alter its content. Figure 4 shows that only two out of the 20 students refrained from opening the file. Out of the remaining 18 who did execute the worm, 16 never noticed a change in the file, while two spotted inconsistencies but ran it nevertheless.

Although a sample of 20 students may not be large, these participants were highly technical individuals, enrolled in computing programmes, ranging from information systems to security and forensics. Yet, in their majority they were deceived into executing the worm. From this point on, for ethical reasons, the rest of the experimental process was carried out in an isolated environment without further participant input, with the assumption that the worm had been executed. Tables 1 - 5 show the impact observed in different experimental scenarios. We used two popular protection systems, McAfee VirusScan Enterprise 8.7.0i and Microsoft Security Essentials with the latest updates installed. None of the two alerted the user of malicious activity or hindered the success of any of the attacks described here.

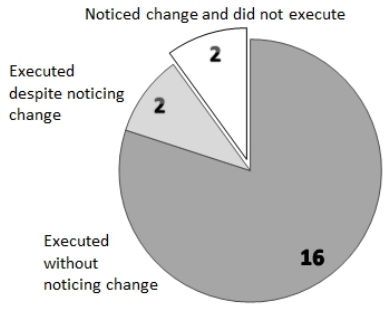

Fig. 4 Experimental results of worm execution

Step	Test Action	Impact
1	Copy worm to local host user profile directory	'csrss.exe' stored in local directory and relevant registry entry is created.
2	Alter 'hosts' file under system directory for local DNS	'hosts' file contains IP - domain name mappings to redirect Google to phishing site.
3	Worm will be executed on XP virtual machine build in current clean state	SugarSync and DropBox cloud applications storage directories should be infected with worm via file masquerading exploit.
4	Copy worm to DropBox and SugarSync Cloud application directories.	Worm impersonates existing file in cloud directory and removes the original.
5	Worm spreads via removable and remote/network media.	Worm impersonates existing file n network storage directory and removes the original.
6	Worm downloads the key logger payload executable from the attacker FTP server.	Keylogger stored locally and registry value for execution on login is created under the current user key space.
7	Worm uploads key logger text file with captured keys to attacker FTP server.	Upload will fail as key logger will not have run and no text file exists.

Table 1 Scenario 1. Windows XP SP3 Blackbox Test - No Anti-virus (No USB infection phase)

Step	Test Action	Impact
1	Execute worm on laptop with USB media attached	Worm and autorun file copied to removable media.

Table 2 Scenario 2. Windows XP SP3 WhiteBox Test - No Anti-virus (No USB spread phase)

Step	Test Action	Impact
1	Worm will be executed on XP virtual machine build in current clean state, with anti-virus installed	Anti-virus does not identify worm running on host.
2	Checking worm functions success	Worm successfully completes all operations without anti-virus intervention or alert.
3	Worm and Keylogger execution on user login	Keylogger and worm successfully run without anti-virus intervention or alert.

Table 3 Scenario 3. Windows XP SP3 BlackBox Test - Anti-Virus Installed

Step	Test Action	Impact
1	Worm will be executed on Windows 7 virtual machine build in current clean state, with anti-virus installed	Anti-virus does not identify worm running on host.
2	Checking worm functions success	Worm successfully completes all operations without anti-virus intervention or alert.
3	Worm and Keylogger execution on user login	Keylogger and worm successfully run without anti-virus intervention or alert.

Table 4 Scenario 4. Windows 7 BlackBox Test - Anti-Virus Installed

Step	Test Action	Impact
1	Instance of Internet Explorer opened and URL www.google.com browsed to	Hosts file redirects DNS mapping to phishing site IP address.
2	Free Download of Google Guard Anti-Virus initiated by hyperlink	Zip file containing scare ware prompts for download.
3	Attempt to use Google search functions and tools	Redirection to phishing site error page.
4	Scareware application executed and application virus check and cleaning functions used	Scareware displays infections, enable cleaning options. On selection redirect to an activation 'attacker' web page prompting for email credentials from the user for activation.
5	User has entered credentials and received activation code, enters wrong code into worm.	Activation unsuccessful and prompts for correct code.
6	User enters correct code	Activation confirmation, complete clean enable.
7	User selects completed clean	Clean confirmation, all options disabled and hosts file cleaned.

Table 5 Scenario 5. Phishing Site and Scareware Testing- (XP Build Anti-Virus Installed)

5 Defence recommendations and future work

Proactive and pre-emptive solutions against social engineering are still at the stage of best-practice suggestions and have not been agreed as standards [5, 6]. The *Signing Seal* technology employed by Yahoo [11] and web application and data security used in RAPPORT [10] help to determine site legitimacy and identity assurance and can be combined to provide a baseline technological solution. Data tagging schemes and enforcement techniques have been proposed as the basis for end-to-end application security in the cloud [13], but they do not take into consideration the human element. Even if a cloud application prevents an external process from accessing the synchronised directory in the local machine, it still does not prevent the user from uploading/downloading a malformed file to/from the cloud application.

What we are missing is a solution that takes into account not only the technical system configurations, but also the users themselves as individuals with, for example, different levels of curiosity or risk aversion in different situations. Protection against semantic attacks may require a hybrid approach, combining technical access control with user conformity, education and training. Current controls recommend "least user rights" and comprehensive security training and education to build up a user's security profile [2], which may be feasible in a business environment, but a home user would be unlikely to offer themselves least user rights or build their own security training policy. For home users, we believe that an online initiative that would involve cloud service providers rewarding users who build their security profile in compliance with best practices, would be beneficial.

6 Conclusions

The aim of this paper was to demonstrate the feasibility of an automated semantic attack in popular cloud storage services. The case study worm exploits the cloud service's reliance on the interface provided to the user by the user's own operating system. It merely requires the user to open the file for the exploit to be complete. Then, the file in the Cloud storage is replicated all over the Cloud infrastructure for that user's account. Since no specific technical vulnerability of the system is targeted and all processes performed in the technical manner they were supposed to, such an automated semantic attack cannot be detected by technical security software. The point demonstrated with this work is that semantic attacks cannot only adapt, but even be automated in the cloud, because they exploit core market drivers behind cloud computing, such as convenience, reduced ownership and technical simplicity.

References

1. Bruce Schneier. Inside risks: semantic network attacks. Communications of the ACM, 43(12), p. 168, New York, NY, USA, December 2000.
2. M.I. Hasan and N.B. Prajapati. An Attack Vector for Deception Through Persuasion Used by Hackers and Crackers. Networks and Communications, pp. 254-258, ISBN 978-1-4244-5364-1, 27-29 December 2009.
3. K. Mitnick and W.L. Simon. The Art of Deception: Controlling the Human Element of Security. ISBN 978-0471237129, Wiley, Indianapolis, USA, 2002.
4. M. Mulazzani, S. Schrittwieser, M. Leithner and M. Huber. Dark Clouds on the Horizon: Using Cloud Storage as Attack Vector and Online Slack Space. Proceedings of the 20th USENIX conference on Security, CA, USA, 10-12 August 2011.
5. Gary Hinson. Social Engeering Techniques, Risks and Controls. The EDP Audit, Control, and Security Newsletter, pp. 32-45, 2008.
6. C. Latze and U. Ultes-Nitsche. How to Protect even Naive User against Phishing, Pharming and MITM Attacks. Communication Systems, Networks, and Applications, pp. 111-116, Beijing, China, October 2007 .
7. Tiantian Qi. An Investigation of Heuristics of Human Judgement in Detecting Deception and Potential Implications in Countering Social Engineering. IEEE Intelligence and Security Informatics, pp. 152-159, New Brunswick, USA, May 2007.
8. T.R. Levine, R.K. Kim, H.S. Park, and M. Hughes. Deception detection accuracy is a predictable linear function of message veracity base-rate: A formal test of Park and Levines probability model. Communication Monographs, 73, 243-260, 2006.
9. Yanpei Chen and Randy H. Katz. Glimpses of the Brave New World for Cloud Security. Feature Article. HPC in the Cloud, 22 February 2011.
10. Trusteer. Rapport OVerview. Retrieved from Trusteer Building Trust Online: http://www.trusteer.com/product/trusteer-rapport, 2011.
11. Yahoo Inc. What is a sign-in seal? Retrieved from Yahoo Security Center: http://security.yahoo.com/article.html?aid=2006102507, 2012.
12. Myles Jordan and Heather Goudey. The Signs, and Semiotics of the Successful Semantic Attack. 14th Annual EICAR Conference, pp. 344-364, Malta, 2005.
13. Jean Bacon, David Evans, David M. Eyers, Matteo Migliavacca, Peter Pietzuch, and Brian Shand.Enforcing End-to-end Application Security in the Cloud. 11th International Middleware Conference, Bangalore, India, Nov. 2010.