Generating randomised virtualised scenarios for ethical hacking and computer security education

SecGen implementation and deployment

Z. Cliffe Schreuders and Lewis Ardern
School of Computing, Creative Technologies and Engineering
Leeds Beckett University
Leeds, UK

***Abstract*—** Computer security students benefit from having hands-on experience with hacking tools and with access to vulnerable systems that they can attack and defend. However, vulnerable VMs are static; once they have been exploited by a student there is no repeatable challenge as the vulnerable boxes never change. A new novel solution, SecGen, has been created and deployed. SecGen solves the issue by creating vulnerable machines with randomised vulnerabilities and services, with constraints that ensure each scenario is catered to specific skills or concepts. SecGen was successfully deployed to generate VMs for a second year undergraduate team module. Future plans are discussed.

Keywords—visualization; exploitation; vulnerabilities; learning environment

I. Introduction

Computer security students benefit from having hands-on experience with hacking tools and with access to vulnerable systems that they can attack and defend. Virtualisation provides a means for legal application of ethical hacking, and the use of virtual machines (VMs) for security education within academia and industry has become widespread [1-3]. For instance, Armitage *et al.* [1] highlight that learning is best achieved when users have privileged access to operating systems with the ability to set-up complex networks using virtual machines.

VMs provide a means for delivering vulnerable targets for ethical hacking, and various VMs can act as targets when learning ethical hacking skills. Metasploitable2 is a popular vulnerable Linux VM, which has many vulnerabilities [4]. Sources such as VulHub provide various vulnerable VMs with different aims and challenges [5].

However, existing vulnerable VMs are static; that is, they do not change and once they have been exploited there is no remaining challenge. This also poses an issue when providing challenges to students, as they are all typically faced with identical scenarios.

This paper presents a unique solution to this problem, by generating VMs of randomised security scenarios.

II. Aims

The aim of this work was to create and deploy a free and open source software (FOSS) method for providing randomised scenarios that contain variations in available services, configuration, and vulnerabilities, while providing control over constraints for the scenarios that are generated.

III. Security Scenario Generator (SecGen)

Security Scenario Generator (SecGen) is a FOSS Ruby application that generates a network of intentionally vulnerable virtual machines (VMs) with randomly chosen vulnerabilities for practising penetration testing techniques.

SecGen has an XML based configuration, which is used to specify constraints for generated scenarios. As illustrated in Figure 1, scenario configurations can specify systems with vulnerabilities, services, and networks. Where details are not provided in the specifications – for example, if the CVE is not specified – these are randomly selected from the available vulnerabilities, services, or networks.

This approach provides a large degree of flexibility in specifying scenarios, while enabling constraints to be used to ensure each scenario is catered to specific skills or concepts. The example SecGen configuration in Figure 1 generates two VMs:

- A remote storage system with random vulnerabilities, while guaranteeing that there will be file storage services, and that there will be a remotely exploitable vulnerability that will result in user-level access that can then be further attacked to gain root access via a privilege escalation attack;
- A VM with a vulnerable webserver that can result in root access.

SecGen processes the configuration files, then uses Vagrant and Puppet to generate and manage the VMs. Vagrant is a system for creating VMs, primarily used by software developers to create and share reproducible and portable development environments [6]. Puppet is a system for remotely managing the configuration of Unix-like and Windows systems [7]. SecGen generates a Vagrant file based on its configuration and uses Puppet manifests to define the services and vulnerabilities to include. SecGen also writes to a report with details of the VMs that have been created.

```
<systems>
	<!-- an example remote storage system, with a remotely exploitable
	vulnerability that can then be escalated to root -->
	<system id="storage-server" os="linux" basebox="puppettest" url="" >
		<vulnerabilities>
			<vulnerability privilege="user" access="remote"
			type="ftp" cve=""></vulnerability>
			<vulnerability privilege="user" access="remote"
			type="" cve=""></vulnerability>
			<vulnerability privilege="root" access="local"
			type="" cve=""></vulnerability>
		</vulnerabilities>
		<!-- secure services will be provided, if matching insecure
		ones have not been selected -->
		<services>
			<service type="ftp"></service>
			<service type="smb"></service>
			<service type="nfs"></service>
		</services>
		<networks>
			<network name="homeonly"></network>
		</networks>
	</system>

	<!-- an example remote web server, with a remotely exploitable root
	vulnerability -->
	<system id="web-server" os="linux" basebox="puppettest" url="" >
		<vulnerabilities>
			<vulnerability privilege="root" access="remote"
			type="www" cve=""></vulnerability>
		</vulnerabilities>
		<networks>
			<network name="homeonly"></network>
		</networks>
	</system>
</systems>
```

Fig. 1. Example scenario configuration file.

IV. Deployment

SecGen was used to generate virtualised scenarios for a second year undergraduate Team Project module. Students worked in teams to conduct security assessments. Teams were provided with a detailed scenario including organisation details and aims, security policy documents, physical locations of premisses, and VMs representing their computer systems. Three SecGen scenario definitions were used as a base to generate VMs for the eight teams, additionally a Website theme was created for each team to further customise each scenario, and these were used to generate five to six VMs per team. Deployment was successful, and indiscernible to students.

V. Discussion

SecGen is capable of overcoming the issue of static security scenarios, and can generate systems which contain random vulnerabilities, while still having meaningful constraints for learning or testing specific skills. This makes it suited to generating scenarios for students, to mitigate the chances of collusion or plagiarism, and also to increase the challenge on repeat tasks. The scenarios can be regenerated to continue to challenge the user since they won't know what vulnerabilities are on the system, allowing them to further progress their penetration-testing skills.

The main limitation of the current implementation is the number of different vulnerabilities the system currently has to choose from. Due to the time-frame to develop the project, only a limited number of vulnerabilities were created. This limits the system since in it's current state students would not be challenged after attempting a few scenarios because they would eventually run out of unique vulnerabilities to attack (especially given the number of VMs involved in each scenario for the Team Project module).

SecGen is released as free and open source software; although some vulnerability definitions have been kept unreleased, until the bank of vulnerabilities is larger, at which point these will also be released publicly.

VI. Future work

Further vulnerability definitions will be developed. Also, more operating systems will be added to the OS selection such as Windows, Mac, and other distributions of Linux, which will help to employ a wider range of exploitation techniques. There are also plans to develop the capability for SecGen to generate Websites that contain randomly injected Web vulnerabilities; such as XSS and SQLi flaws. More "loot" content and themeing will also be added to create more complete and convincing scenarios. Finally, it would be desirable to develop a scoring system: for example, marks for submitting flags, loot, or CVEs, which would make SecGen particularly well suited for automated assessment marking and capture the flag (CTF) competitions.

VII. Conculsion

In summary, a new system for generating VMs for security education has been implemented and deployed. A unique feature of this system is the ability to generate meaningful scenarios for learning security and ethical hacking concepts, which are randomised, meaning that separate students or users get appropriate learning experiences, but with separate challenges. SecGen has successfully been used to generate VMs for team project security assessments. There are plans to continue development, and seek development collaborations.

References

[1] Armitage, W.D., Gaspar, A. & Rideout, M. (2007) Remotely Accessible Sandboxed Environment with Application to a Laboratory Course in Networking. In**:** Proceedings of the 8th ACM SIGITE Conference on Information Technology Education. SIGITE '07. New York, NY, USA, ACM, pp.83–90.

[2] Hu, D., & Wang, Y. Y. (2008, April). Teaching computer security using Xen in a virtual environment. In Information Security and Assurance, 2008. ISA 2008. International Conference on (pp. 389-392). IEEE.

[3] Hu, J., Meinel, C., & Schmitt, M. (2004,). Tele-lab IT security: an architecture for interactive lessons for security education. In ACM SIGCSE Bulletin (Vol. 36, No. 1, pp. 412-416). ACM.

[4] Moore, H.D., Metasploitable 2 Exploitability Guide, http://r-7.co/Metasploitable2

[5] VulnHub: Vulnerable by Design, https://www.vulnhub.com/

[6] Hashimoto, M. (2013). Vagrant: Up and Running. O'Reilly Media, Inc.

[7] Walberg, S. (2008). Automate system administration tasks with Puppet. Linux Journal, 2008(176), 5.