Chad Ballay

CYBR410

10/12/2020

Week 7 -Project 7-1

1. **What are the first steps that you would take to test the sites for SQL injection vulnerability?**

First step would be ascertaining the details of the tech stack we are up against. Is it a single node or multiple nodes? Managed service or self-hosted? What version and what patches? Is there a load balancer involved? What are the known forms/queries/mechanisms that are already interacting with the DB server?

It’s step one of the Cyber Kill Chain (Hospelhorn, n.d.). Identify the target as clearly as possible with as much resolution of detail as possible. Begin with a focus as being as passive as possible and slowly escalate to active information gathering. At each juncture, try to identify what additional piece of the puzzle do you need and how you’ll get it with as little disturbance to the system as possible.

1. **How might you apply the concept of inferential testing?**

Inferential testing combines two concepts to yield a larger result than either alone would. The first is establishing how does this system deviate from expected behaviors. The first is looking for what is configured out of the norm or behaving out of the norm? Being able to identify how does this system deviate helps to give you insight into the unviewed parts of the system. The second is looking at how typical users use the system. This provides you with a background of how far you can push or pull this system before being noticed. What are the things you can try before your behavior stands out as a signal from the noise. (Inference Attack, n.d.)

Much of inferential testing isn’t actually touching or manipulating the parts of the system you are really after. It instead is giving you this insight by how other parts of the picture can be used to reflect or infer that insight. Careful deduction and reasoning is required for this stage.

1. **What is your strategy for identifying dangerous source code now and far into the future?**

There is no single silver bullet for this. It would require a comprehensive set of items to be implemented and maintained.

One of the more intriguing ones would be embedding honeytokens in the source or in the database tables. The idea being that these items should never be accessible in the outside world. An email address that should never get any mail. Or a URL that should never be visited. Or a unique string of text that can be searched for on various forums. The point being that if these items are ever triggered then you know that you have a breach. ( Canarytokens.org - Quick, Free, Detection for the Masses, n.d.)

1. **What suggestions would you offer TJRiggings in reference to their web clients?**

You will fail at some percentage of the requirements for each of your clients. If and when that occurs your client will also not be putting their best face forward. Managing that relationship while protecting TJRiggings requires a deft touch of compassion and acumen. Build the failsafes up front to smooth these moments. Build a culture of performing post-mortems. Communicate transparently when resolving an issue. Establish non-negotiables that if you do end up having to negotiate away you have black and white what they were negotiated away as.

And for heaven’s sake, demand observability and resiliency testing as a pre-condition for a completed project (Burn, n.d.). Being able to withstand crippling failures across multiple levels of the tech stack is feature that many people only appreciate after it is too late. Knowing what and how the system is performing is always step 1. Having that information will help to remove the ego and finger pointing.

# Works Cited

*Canarytokens.org - Quick, Free, Detection for the Masses*. (n.d.). Retrieved from Thinkst: https://blog.thinkst.com/p/canarytokensorg-quick-free-detection.html

Burn, J. (n.d.). *The Importance of Resilience Testing and Observability*. Retrieved from The New Stack: https://thenewstack.io/the-importance-of-resilience-testing-and-observability/

*First-Mover Advantage*. (n.d.). Retrieved from WIkepidea: https://en.wikipedia.org/wiki/First-mover\_advantage

Hospelhorn, S. (n.d.). *What is The Cyber Kill Chain and How to Use it Effectively*. Retrieved from Varonis: https://www.varonis.com/blog/cyber-kill-chain/

*Inference Attack*. (n.d.). Retrieved from Cybary: https://www.cybrary.it/glossary/i-the-glossary/inference-attack/

*Infrastructure as Code*. (n.d.). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Infrastructure\_as\_code

Sgobe, T. (n.d.). *Installation and Oracle Database Creation in Silent Mode*. Retrieved from EzDBA - BLOG: https://ezdba.wordpress.com/2016/09/01/installation-and-oracle-database-creation-in-silent-mode/

Shroff, A. (n.d.). *Oracle Developer's blog*. Retrieved from Infrastructure as Code using Terraform on Oracle Developer Cloud: https://blogs.oracle.com/developers/infrastructure-as-code-using-terraform-on-oracle-developer-cloud