JHU SU20 IDS Module 3 Assignment

Audrey Long

06/14/2020

Objective

You should submit your response by clicking the "Module 03 Assignment" link above and uploading a file containing your written response.

- What platform information sources are available to an IDS that would contain observables for the attack on a 4758 described in Chapter 18 - API Attacks.pdf ? Why? Hint: Section 18.2.2 provides some good insight.
- What type of attack would not contain observables/not be visible to any (generic) HIDS? Why not?
- Please submit your assignment in either pdf or Microsoft word format. 1 1/2 to 2 pages of content is sufficient

API attacks on 4758

With software in today's world a lot of software development is done using micro services and APIs which are also stored and acted upon in the cloud this makes re-usability and provides interfaces for many different services and applications to interact with your website or service provided. This is very ideal in the perspective of a software engineer, but when you look at all the freedom and information exchanges with a security perspective many things can be intruded, listened upson, changed, used, and attacked in many different ways.

For this short essay we will focus on a specific API attack on the 4758. For a bit of background the IBM 4758 is a secure PCI cryptographic processor which is implemented on a high security tamper resistant programmable PCI board. Specialized cryptographic electronics, microprocessor, memory, and random number generators housed within a tamper-responding environment provide a highly secure subsystem in which data processing and cryptography can be performed. An exploit of this system points to the generation of check values for keys and a key identifier calculation which encrypted a string of zeros under the key, which opens up an attack otherwise known as the birthday attack which is also known as a time memory tradeoff attack.

This article talks about the following steps that can be taken to break a DES key:

1. Generate a large number of terminal master keys, and collect the check value of each.
2. Store all the check values in a hash table.
3. Perform a brute force search, by guessing a key and encrypting the fixed test pattern with it.
4. Compare the resulting check value against all the stored check values by looking it up in the hash table (an O(1) operation).

The attacker could generate 2^16 target keys then try to crack the key with 2^40 chance of getting it right, this is known as a meet in the middle attack. With the information we know about how to crack the keys on this device we now know that the attacker would not need physical access to the device. With the implementation of HIDS we can detect the addition and deletion of files on the system for when the attacker does all of the key creation, we can generate logs to indicate that a DES. For this type of attack a network based intrusion detection system could prevent tampering with messages before reaching the host based system in a more efficient manner than the host based system could find whereas the host based system is really only useful for file integrity, and logging malicious events.

We also know that the type of attack used to extract the key is called a meet in the middle attack which refers to a generic space time tradeoff cryptographic attack against encryption schemes that rely on performing multiple encryption operations in sequence. Meet in the middle attacks generally happen when a communication between two systems is intercepted by a bad actor and injected with malicious content and sent off.

Attacks without Observables

In this section we will briefly discuss some attacks that would not contain observables and would not even be visible to a host based intrusion detection system. Essentially HIDS operates on information that has been collected on the system which can be analyzed and determines file integrity, and can detect bad processes and users on the system. According to some light research there are a number of techniques that attackers are using to evade intrusion detection systems and leave behind difficult to find footprints in the making. One of the main evasion techniques is to change patterns to avoid evasion, since a HIDs generally relies on “good pattern matching” to detect something bad by changing the packet payload data slightly the system might miss the bad action completely.

Application hijacking/ application layer attacks can completely bypass HIDS if done correctly. Essentially this attack enables many different forms of evasion, for example many applications deal with media such as images which can come in the form of some kind of compression. For example on the HIDS we may expect a compressed file with some extension on the network but the contents inside once executed contain malicious code. These types of attacks can entirely be executed within the compressed data which makes it tricky for an HIDS to detect the file format for signatures.

References

https://en.wikipedia.org/wiki/IBM_4758 [1]
https://blackboard.jhu.edu/bbcswebdav/pid-7971913-dt-content-rid-18216787_2/courses/EN.695.442.81.SU17/Module%2004/Chapter%2018%20-%20API%20Attacks.pdf [2]
cl.cam.ac.uk/~rnc1/descrack/ [3]
https://en.wikipedia.org/wiki/Meet-in-the-middle_attack [4]
https://www.globalsign.com/en/blog/what-is-a-man-in-the-middle-attack [5]
https://oa.mo.gov/sites/default/files/CC-HostBasedIDS040303.pdf [6]
https://en.wikipedia.org/wiki/Intrusion_detection_system#Evasion_techniques [7]
https://def.camp/wp-content/uploads/dc2015/tudordamian-idsevasiontechniques-151123083756-lva1-app6892.pdf [8]