Using Botnets to provide security for safety critical embedded systems - a case study focused on UAVs

Fernando Augusto Garcia Muzzi[1], **Paulo Rogério de Mello Cardoso**[2], **Daniel Fernando Pigatto**[3], **Kalinka Regina Lucas Jaquie Castelo Branco**[3]

[1]FATEC Lins – Faculdade de Tecnologia de Lins – FATEC.
[2]Centro Universitário Senac, SENAC/SP, Brasil.
[3]Universidade de São Paulo – Instituto de Ciências Matemáticas e de Computação, Av. Trabalhador São Carlense 400, Centro, CEP 13566-590, São Carlos, SP, BR.

E-mail: fagmuzzi@yahoo.com.br

Abstract. The use of unmanned aerial vehicles (UAVs) has been growing not only in military applications, but also in civilian. UAVs have enormous potential for use, which mostly still are unexplored. For the use of UAVs in the airspace, not only Brazilian new studies on methods of analysis and technologies should be incorporated into navigation systems, control among others, promoting security mechanisms for these aircraft. Implement security mechanisms using a platform with operating systems and botnet to simulate such attack Distributed Denial of Service (DDoS) in UAVs is an important task when it is aimed at containment and mitigation of attacks on this type of platform.

1. Introduction

Aerial vehicles have increasingly receiving more electronic components over the last years, which are connected through wired and wireless networks and running embedded software. This integration of dedicated computing devices, the physical environment and the network composes a Cyber-Physical System (CPS). CPS has thus become part of common vehicles, accessible to everyone, such as ground and aerial vehicles, including the autonomous ones. These vehicles have processing power increased and a very sophisticated embedded software, adding a big amount of sensors, becoming each day a more complex system. These complex embedded systems, like UAVs (Unmanned Aerial Vehicles), are able to perform lots of missions. If failure events occur, it may be caused losses of human lives and high-value assets, what makes such systems critical embedded systems.

The combination of high mobility and wireless communications in such systems has further increased their exposure to malicious threats and to faults deriving from uncertain connectivity or communication timeliness. Security and real-time operation, like other non-functional requirements, have become harder to fulfill, creating new challenges to such safety-critical embedded systems. This paper addresses the security problem in UAVs, focused in the wireless interface and in the Distributed Denial of Service attack by using BotNets.

2. Sphere: Safety and Security Platform under HAMSTER Architecture

Sphere is the platform for safety & security in HAMSTER architecture [7]. Although the platform may have centralized modules, it is not a centralized platform. Sphere is present in many parts of the unmanned vehicle (UV) according to its necessities and is responsible for information security (the

way it is exchanged, stored, manipulated etc.) and also for health and safety of the overall vehicle and all subsystems that compose the unmanned system. Only a few parts of an UV are properly treated to ensure that all connected modules are authentic and have not been replaced or tampered with by a third party. The current policy adopted by most aircraft manufacturers uses a concept of "Accept all" which trusts in all components embedded in an aircraft. This proposal suggests the assumption of an "Almost Deny All" approach, which denies the authenticity of all mechanical components and peripherals attached to the vehicle until the opposite is proved, which may result in safer vehicles.

The categorization of every module is therefore crucial for such a new security model to be applied to UV. There are various peripheral devices embedded in an UV that require different levels of security, which leads to the necessity of a module categorization according to the criticality of their performed functions. The Sphere proposal suggests the modules categorization into primary, secondary, and so on, according to necessity.

Primary modules are those considered essential components for the UV to operate, to be aware of its location and to be able to perform an emergency abort operation safely, even when the mission was not entirely concluded. An autopilot (focus of this paper), a GPS receiver, and barometric/inertial units are examples of modules classified as primary, since they might cause a big lost if in failure state. In contrast, modules not considered as essential functions to the UV are classified as secondary modules.

Whether abnormal behaviors are detected in any secondary module, the operation of the primary components of the UV is not affected and the secondary module that presented the abnormality should be disabled or isolated. It implies that all primary modules must be authenticated before the operation begins. However, the secondary modules do not necessarily need an authentication before the mission execution. The creation of access profiles for modules is another concept associated with the proposal of authentication. As a mission is assigned to the UV, it must go through an authentication process, which assigns different access permissions to the UV modules. Such concept is similar to the one used in modern operating systems where an administrator user is allowed to install and uninstall software with no restrictions, unlike a visitor user who has access to the programs, but is not allowed to install/uninstall them. Applied to UV, such concept adds a layer of security that allows blocking the use of selected modules by specific users. Such specification is intended to prevent unauthorized access. Even if there is a single effective user of an UV, no other user (an attacker or not) will have privileged access to modules or their information.

Figure 1 presents the Sphere main modules. Central Security Unit (CSU) is responsible by modules authentication and "health" verification. The access policy assigned to each user must use cryptographic algorithms suitable for embedded or real-time sensitive environments. Several experiments have been carried out regarding security for critical embedded systems [4, 5].

To protect the UV against attacks coming from malicious components, Sphere implements strict security policies. It is necessary to ensure that all modules are authentic, so if one of them fails or presents an abnormal behavior, the others must not communicate with it. Furthermore, such policies must be applicable even during vehicle operations, considering that external factors may affect the components behavior e. g. climate or weather changes. During the vehicle start up, a mutual authentication phase should occur with CSU. It checks the database credentials of all modules, their criticality, and even if there is any access restriction. There is also the possibility of deciding whether a module should be initialized or not during the verification stage. The following steps will be authentication and exchange of encrypted messages to establish a secure channel for communication among modules and CSU. After such handshake, three situations are expected:

1. The module that is trying to authenticate and CSU have not been tampered with;
2. The module has been tampered with and therefore has not been authenticated:
    a. If it is a module of primary type, the UV must not operate;
    b. If it is a module of secondary type, communication with it must be interrupted and a notification be sent to a control station.
3. The module that is trying to authenticate may notice that CSU is not authentic, and must notify other components about it.

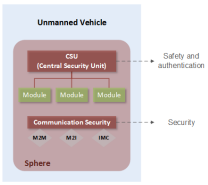

Figure 1. Composition of Sphere, the Safety & Security platform on HAMSTER architecture.

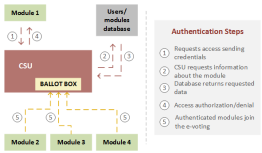

Figure 2. Modules and CSU authentication steps. The "Ballot box" is used by every other system modules for authenticating CSU through e-voting protocols.

From the point of view of communication security, an ideal situation would be if all modules could authenticate with others. However, this method would cause a system overload, since the increase of modules in the aircraft would cause an exponential increase in the number of exchanged messages. To solve such problem there exist the e-voting protocols [6]. Such model can be further expanded according to the needs of the UV, including a negotiation mediated by CSU to create a secure channel of communication among modules. A graphical representation of processes performed during the authentication module with CSU can be seen in Figure 2.

3. Threats to Unmanned Aerial Vehicles

In the scope of this paper three vulnerabilities related to vehicular wireless communication may be mentioned [1]: operating wireless network interfaces to DoS (Denial of Service) attacks and espionage (information theft); malicious code injection into embedded devices present in the vehicle; and induction to incorrect behavior when in cooperation with other vehicles.

Security is a big concern. In [2] the authors presented a UAV system cyber-security threat model, which can be seen in Figure 1. Note that there are several open issues related to communication. Open networks could be easily hacked by malicious entities. Studies show that using BotNets we can stop the communication, and using Botmaster to start the attack with bots to an UAV through vulnerabilities in existing Wi-Fi networks, we can identify vulnerabilities in communication and communication link, important to check for possible threats and stop attacks [2]. Using a real botnet is possible to attack an UAV autopilot and detect all the standard behavior. With the information will be possible to provide countermeasures to stop the attack. Based in Figure 3, we will focus on the attack presented in Figure 4. All these process is integrated with Sphere.

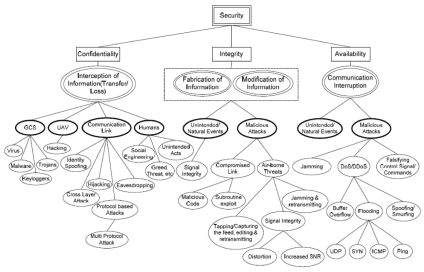

Figure 3. UAV system cyber-security threat model by [2].

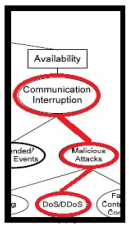

Figure 4. Attack to be analyzed.

4. Results and Discussions

The simulation platform is illustrated in Figure 5 and shows the network infrastructure that contains virtual machines and bots, IRC server for sending botmaster commands to the bots (hosts) to start the attack on a specific target in which is used an autopilot board to simulate DDoS attack on a UAV. We make an attack on the UAV and we detect patterns of behavior by the attack, in order to verify mistakes and to develop containment mechanisms and mitigation of attack.

The first step that we realize to model the DDoS attack was the specification using an extension tool called UML*intr* proposed by Hussein and Zulkernie [4]. But we note that the UML*intr* is not specified for this attack type making necessary the addition of new classes for the characters of DDoS

attack can be represented with all its features. In Figure 6 we present a modeling of a DDoS attack using the UML*intr* extended to allow the DDoS specification (detailed in Figure 7).

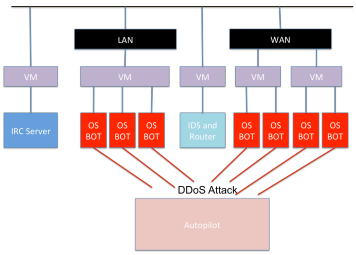

Figure 5. Attack simulation platform.

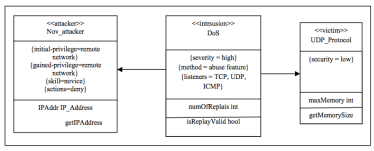

Figure 6. Modeling a DDoS attack.

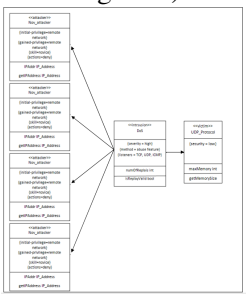

Figure 7. Representation of a DDoS attack elements.

The processes had stalled instantly after the attack. Memory usage has not changed, but the processing was very high utilization rates. At different intervals were given three attacks on the "victim" and the CPU utilization, which left the Idle state (idle) 100 % and reached the level of 0% Idle. We measured the capture packets received by the network, which recorded a flood of UDP packets the type attack used for the DDoS. The percentage of UDP packets was very high during the attack period, representing 99.8% all traffic. Was possible see that when the attack was completed the network interface still responding to requests, but at a lower level of saturation, which proves that even after the effective attack, the victim continues to respond the actions of the attack. With these results, its also possible see the complexity involved in this type of attack and notes the efficiency of the same when struck the victim. Take these in account it is necessary to evaluate DDoS attack to create defense mechanisms.

5. Conclusion

The main objective of this paper was to present a platform to perform a DDoS attack to an UAV to find a way to mitigate this kind of attack. As first results we obtained a DDoS attack modeling and specification. We could also analyze the effects caused by this kind of attack. This is an important step towards to find a way of mitigating such attack. We aim to use the botnet platform to mitigate DDoS attacks in a future work.

Acknowledgment

The authors would like to thank the institutions where they conduct their research for supporting this project and also the financial support provided by CNPq, CAPES and FAPESP (2014/13713-8).

References

[1] Wyglinski, A. M., Huang, X., Padir, T., Lai, L., Eisenbarth, T. R., & Venkatasubramanian, K. (2013). Security of Autonomous Systems Employing Embedded Computing and Sensors. IEEE Micro, 33(1), 80–86.
[2] Javaid, A. Y.; Sun, W.; Devabhaktuni, V. K.; Alam, M.. Cyber Security Threat Analysis and Modeling of an Unmanned Aerial Vehicle System. Homeland Security (HST), 2012 IEEE Conf. on Technologies.
[3] Hussein, M.; Zulkernine, M. UMLintr: A UML for Specifying Intrusion**.** In: 13th ANNUAL IEEE International Symposium and Workshop on Engineering Based Systems, 27-30, 2006, Germany. p. 279-88.
[4] V. Schoaba, F. E. G. Sikansi, D. F. Pigatto, K. R. L. J. C. Branco, and L. C. Branco, “Digital Signature for Mobile Devices: A New Implementation and Evaluation,” International Journal of Future Generation Communication and Networking, vol. 4, pp. 23–36, 2011.
[5] D. F. Pigatto, N. B. F. D. Silva, and K. R. L. J. C. Branco, “Performance Evaluation and Comparison of Algorithms for Elliptic Curve Cryptography with El-Gamal based on MIRACL and RELIC Libraries,” Journal of Applied Computing Research, vol. 1, no. 2, pp. 95–103, Feb. 2012.
[6] H.-T. Liaw, “A secure electronic voting protocol for general elections,” Computers & Security, vol. 23, no. 2, pp. 107–119, 2004.
[7] D. F. Pigatto, L. Goncalves, A. S. R. Pinto, G. F. Roberto, J. F. R. Filho, and K. R. L. J. C. Branco, “HAMSTER - Healthy, mobility and security-based data communication architecture for Unmanned Aircraft Systems,” in 2014 Int. Conference on Unmanned Aircraft Systems (ICUAS). IEEE, May 2014, pp. 52–63.