2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE)

A Proxy-Based Solution for Securing Remote Desktop Connections in Mission-Critical Systems

Ron Bitton, Clint Feher, Yuval Elovici, Asaf Shabtai
Dept. of Software and Information Systems Engineering
Ben-Gurion University of the Negev
Beer-Sheva, Israel
{bitton, feher}@post.bgu.ac.il, {shabtaia, elovici}@bgu.ac.il

Gaby Shugol, Raz Tikochinski, Shachar Kur
Astronautics C.A. ltd
{g.shugol, r.tikochinski, s.kur}@astro.co.il

***Abstract*— Remote desktop protocols (RDPs) are used for connecting and interacting with computers remotely. In recent years, we have witnessed a number of vulnerabilities identified in two widely used remote desktop implementations, Microsoft Remote Desktop and RealVNC, that may expose the connected systems to a new attack vector. Such vulnerabilities are particularly concerning when it comes to mission-critical systems in which a client device with a low trust level connects to the critical system via a remote desktop server. In this preliminary study we propose a proxy-based solution that applies various modules, each of which mitigates a different type of threat, in order to secure remote desktop connections used in mission-critical systems.**

Keywords*— *remote desktop, proxy, remote code execution, malware, mission-critical system

I. INTRODUCTION

Remote desktop protocols (RDPs) are commonly used for connecting and interacting with computers remotely. These protocols usually consist of two software components: a client and a server. The server component runs on the remote computer and shares its desktop (i.e., screen) with the client component which runs on an end user device. The client component allows the user to control the remote machine by sending the server I/O events (i.e., pointer/mouse clicks and keyboard presses). Primary reasons for using remote desktop capabilities are: (1) connecting to machines that are not equipped with I/O units (i.e., keyboard and monitor); (2) connecting to machines that are less physically accessible (such as machines in server rooms or in a cloud); and (3) replacing costly (and proprietary) I/O devices with inexpensive and more readily available consumer devices such as laptops and PDAs.

Two widely used remote desktop implementations are the cross-platform, remote framebuffer update protocol (RFB) [12] implemented by RealVNC [11] and the Microsoft Remote Desktop Protocol (MS-RDP) for the Windows operating system [10]. Both, MS-RDP and RealVNC contain authentication and encryption modules to ensure a high level of integrity and confidentiality between the remote desktop client and server components.

In recent years, a number of vulnerabilities have been identified in MS-RDP and RealVNC components. These vulnerabilities can be exploited by an attacker, allowing the attacker to execute code remotely or bypass the remote desktop authentication mechanism. Both cases may result in the attacker's complete control of the system (administrator privileges); similarly, such attacks could prevent legitimate users from connecting the system. Thus, the use of remote desktop applications may expose the connected systems to a new attack vector. This concern is increased when it comes to a mission-critical system in which a client device with a low trust level connects to the critical system via a remote desktop server.

In this study, we propose a proxy-based solution that acts as a man-in-the-middle between the remote desktop client and server applications. The proxy applies various modules, each of which mitigates a different type of threat, in order to secure remote desktop connections in mission-critical systems.

The reminder of the paper is organized as follow. Section II presents a threat analysis for RDP applications. In Section III we present a case study of the electronic flight bag (EFB), demonstrating the use of the RDP in an aircraft's mission-critical system. Section IV presents the general architecture of the proposed RDP proxy. Section V provides a review of related works, and finally, Section VI concludes the paper and presents future work directions.

II. REMOTE DESKTOP THREAT ANALYSIS

In order to properly design a proxy-based solution we analyzed and identified the following threats. These threats can be implemented via the infected device of a legitimate user or via the device of an attacker masquerading as a legitimate user.

A. *Code execution*

Exploiting a vulnerability that exists within the RDP server component may lead to code execution by an unauthorized entity (e.g., via the remote desktop client). An attacker exploiting a vulnerability will be able to run malicious code on the server and may even gain complete access to the critical system (e.g., CVE-2015-2373, CVE-2013-1296, and CVE-2009-1133).

B. *Escalation of privileges*

The remote desktop server component is often a high privileged service (i.e., operates with administrator/root privileges). Therefore exploiting a vulnerability in this service may escalate the privileges of an attacker from a standard user to an administrator/root user (e.g., CVE-2016-0036, CVE-2013-6886, and CVE-2015-2473).

C. Authentication bypass

Although remote desktop protocols implement an authentication phase, an attacker could try connecting to the critical system using an unauthorized device by spoofing client credentials (e.g., CVE-2015-2472). In addition, a malicious entity could use stolen credentials (obtained by applying social engineering techniques). Furthermore, bypassing the authentication component could be achieved by exploiting vulnerabilities that exist within the RDP server (e.g., CVE-2016-0019, CVE-2014-6318, and CVE-2006-2369). In all of the above scenarios the attacker may gain complete access to the system.

D. Denial-of-service (DoS)

In cases where RDP is the only available means for connecting to the system (e.g., when the system has no I/O units or is not physically accessible), the result of exploiting vulnerabilities in the RDP server component (e.g., CVE-2015-0079, and CVE-2012-0152) could prevent legitimate users from establishing a connection to the system.

E. Spoofing

A mission-critical system usually has a high level of trust. Connecting to such systems using a device with a lower level of trust via remote desktop application may expose the system to a new attack vector. A malicious application residing within the end user device could send false commands on behalf of the user or alter the commands performed by the user in the critical system. This threat is especially relevant when using devices owned by the end user (i.e., "bring your own device" policy).

III. EFB Case Study

An electronic flight bag (EFB) is a highly trusted mission-critical server used by pilots in civil aircrafts. The EFB provides important information to pilots during the flight, such as airport charts and maps, flight operation manuals, weather information, checklists, and performance calculators. The EFB is connected to most of aircraft's avionic computers and sensors, such as navigation and maintenance equipment, in order to display the information to the pilots using dedicated avionic applications installed in the EFB. Using the EFB as part of the aircraft's flight management system helps airlines improve flight safety. The EFB primarily consists of a computational unit (i.e., EFB server) and a display unit (i.e., touch-screen monitor) as presented in Figure 1.

Due to its safety-critical operation, the EFB requires certification to meet the DO-178 safety requirements of the Federal Aviation Administration (FAA) [14]; certification involves complex processes of development, documentation, and testing which are accompanied by professional representatives of the FAA.

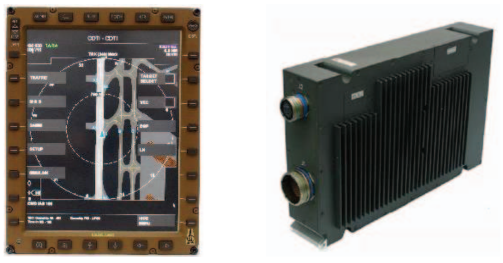
Figure 1: EFB monitor and server.

Since the EFB display unit is an expensive device, EFB manufacturers have considered alternative solutions including replacing the existing custom display, and the use of a support connection to tablets via remote desktop protocols such as VNC. In this case, the connection and necessary security features re made possible using a dedicated router; the EFB server is connected to the router via an Ethernet connection, and the tablet is connected to it using a wireless connection as shown in Figure 2. Because these tablets, provided by the airline companies, are not typically hardened and are used by the pilots as personal devices (similar to the *bring your own device* case), the EFB specifically, and consequently other aircraft systems, are exposed to the risks described in Section II.

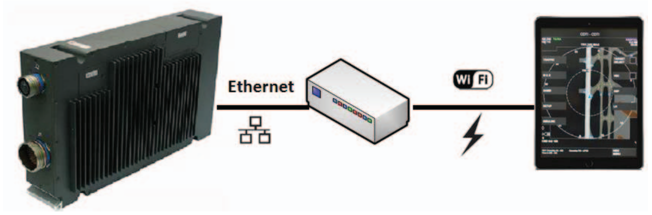

Figure 2: A tablet connecting to an EFB server through a VNC using a dedicated router.

The proposed remote desktop proxy architecture can protect the EFB server from attacks that can be initiated via an untrusted tablet on the remote desktop application. In this case, the proposed proxy can be implemented as a proxy-firewall inside the router.

IV. General Architecture of the RDP Proxy

As presented in Figure 3, the proxy solution is designed as a system that provides three modules of protection to the remote desktop application: device verification, user verification, and code injection detection. Additional security modules (e.g., denial-of-service protection) may be implemented and added at any time.

A. Device verification

The first goal of the proxy is verifying that the end user device connected (e.g., the tablet in the EFB use case) is a device known to the system and authorized by the system.

As presented in Section II, an adversary can authenticate to the system by stealing the credentials of a legitimate user, or can manipulate the RDP authentication components which may result in bypassing the authentication. Since device identity information such as the MAC address can be spoofed, there is a need for an alternative method for verifying the device identity during and after the authentication phase.

Thus, the Device verification module is to continuously monitor the interaction between the end user device (RDP client) and the system (RDP server), verify the identity of a connected end user device (e.g., tablet), and prevent unauthorized devices from connecting to the remote machine (e.g., EFB server).

One approach for achieving this goal is applying machine learning-based anomaly detection techniques. This can be done by combining *contextual information* with behavioral patterns of the end user device that are extracted from the *network traffic* transmitted between the device and the system, and matching these patterns with a pre-learned profile of the specific device, in order to detect abnormal behavior within the network communication. Examples of contextual attributes that can be utilized are the day and time of connection, the role of the user in the organization, or the stage of flight (i.e., takeoff, landing, and cruising altitude) in the EFB use case.

Devices such as tablets include a diverse set of applications, including built-in applications provided by the vendor and others installed by the user. The proper functionality of some of the applications necessitates accessing a server for information and updates, generating various traffic (including DNS, TCP, and UDP traffic). If available, traffic that is not related to the direct RDP communication between the device and the system can therefore be utilized for profiling each end user device. For example, in the EFB use case both the EFB server and tablet are connected to the router. This setting allows the router to capture *all* traffic produced by the device, including applications' DNS requests, and not only the traffic generated by the remote desktop application.

This security module should be capable of extracting relevant data from the appropriate source of information; furthermore, it should be able to generate, update, and apply the devices' behavioral profiles and eventually determine whether the identity of the declared device is in fact the stated one. In cases in which the similarity level is not high enough, an alert may be issued.

B. Detecting code injection

As presented in Section II, code injection is a primary attack vector related to the integration of remote desktop protocols in mission-critical systems. Therefore, the second security module utilizes the remote desktop network traffic in an attempt to detect anomalies, such as code injection events, and prevent the EFB server from being exploited.

To better understand how to protect against code injection in remote desktop protocols we analyzed the CVEs described in Section II and derived following insights: (1) a packet containing an injected code consists mostly of non-valid data and therefore looks different from a standard packet; and (2) in most of the cases in which a RDP session leads to a code injection event, the sequence of protocol events is abnormal. Therefore, in order to identify code injection events, we suggest combining a point anomaly detector with a sequence-based anomaly detector.

The *point anomaly detector* analyzes each RDP packet separately in order to detect an abnormal packet (i.e., a packet containing injected code). This component can utilize attributes extracted from the TCP header (e.g., packet length, packet source, segmentation flag, etc.) and the TCP payload (e.g., distribution of bytes within the packet payload) to calculate an anomaly score for each RDP packet. The *sequence-based anomaly detector* accumulates TCP packets into sessions and extracts the sequence of RDP events. By applying a sequence-based anomaly detection algorithm, this component is able to derive an anomaly score for every RDP TCP session.

Note that a major challenge associated with implementing this security module is the ability to identify the type of RDP event by processing the TCP traffic.

C. User verification

The user verification module consists of two components. The first component aims to distinguish between a human user and a *bot* (i.e., a malicious code) that exists in the low trust machine and sends false commands to the server on behalf of the user. The second component aims to discriminate between various users and provide continuous verification of the authenticated user. To accomplish their aims, both of the components can utilize the pointer's movement and keyboard events transmitted by the client to the server, together with the screen bitmap updates transmitted by the server to the client. Those features are the input to a sequence mining model that results in a user-specific profile that can be used for user verification.

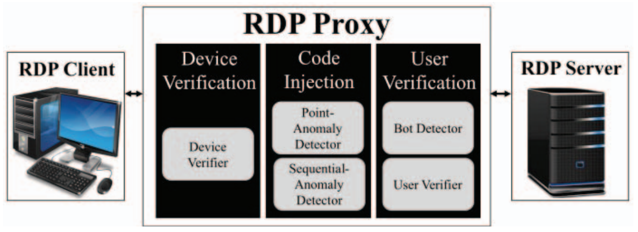

Figure 3: General architecture of the remote desktop proxy.

V. Related Works

Device verification (also known as device fingerprinting) is a means of continuously confirming that the proclaimed identity of the device is the actual (legitimate) identity. Verification is performed by generating device-specific signatures. Device identification is important for preventing node forgery and insider attacks in wireless networks, as various device identifiers, such as the MAC address and IMEI, can be changed and forged by software. Works in this domain can be classified as passive and active methods. Passive methods identify the device by processing various sources of information, such as analog hardware signals [1] [2], network features (e.g., frame size), and timing analysis (e.g., frame transmission time and inter-arrival time) [3] [4]. Active methods use different challenges to identify the devices by their response to malformed and non-standard frames [5]. In this study the proposed device identification module implements a passive method, generating device signatures based on application DNS requests and their time characteristics. An additional active device identification module can be added to the proxy to improve identification accuracy.

Prior works related to user verification presented methods for profiling users based on their mouse and touchscreen dynamics by analyzing information extracted on the end user device [13] [14]. In this study we propose performing user identification by analyzing the RDP network traffic and extracting information that is related to actions performed by the user (mouse/pointer movement and keyboard strokes), as well as screen update events resulting from these actions.

Previous studies related to the identification of code injection in network traffic mostly used point anomaly detection. In [15] the authors presented a method which models the normal application payload in order to detect abnormal payloads (i.e., a payload contains malicious code). In [16] the authors presented a framework which detects binary shell-code within network payloads and run it on a monitored environment. In this study we propose an integrated method, which combines point and sequential anomalies in order to better detect code injection events.

To the best of our knowledge, no solution has been proposed based on a security proxy that provides protection to remote desktop connections. The only related study identified suggests a proxy-based architecture for security auditing of remote desktop events, together with a single sign-on (SSO) login mechanism, which provides a single authentication module for several remote desktop applications (e.g., MS-RDP, VNC and X-Window) [6].

VI. Conclusions And Future Work

In t`his study we propose a proxy-based security solution for remote desktop implementations. The proposed solution aims at securing the remote connection protocol and thereby protecting the actual systems from cyber-attack. The proposed framework can be extended to other applications that can be used for remotely connecting and interacting with mission-critical systems, such as HTTP connections via browsers or VPN connections. In future work we plan to provide a detailed design of the proxy modules and detection algorithms, implement the proxy, and evaluate its performance within a test environment that will be implanted for this task. In addition, we plan to utilize an additional set of features that might be extracted from the end user devices and shared with the proxy, such as the location and accelerometer measurements available in tablets, which were previously used for user verification.

VII. References

[1] Brik, V., Banerjee, S., Gruteser, M., & Oh, S. Wireless device identification with radiometric signatures. 14th ACM international conference on Mobile computing and networking, 2008.

[2] Nguyen, N. T., Zheng, G., Han, Z., & Zheng, R. Device fingerprinting to enhance wireless security using nonparametric Bayesian method. INFOCOM, 2011.

[3] Desmond, L. C. C., Yuan, C. C., Pheng, T. C., & Lee, R. S. Identifying unique devices through wireless fingerprinting. 1st ACM conference on Wireless network security, 2008.

[4] Neumann, C., Heen, O., & Onno, S. An empirical study of passive 802.11 device fingerprinting. 32nd International Conference on Distributed Computing Systems Workshops, 2012.

[5] Bratus, S., Cornelius, C., Kotz, D., & Peebles, D. Active behavioral fingerprinting of wireless devices. 1st ACM conference on Wireless network security, 2008.

[6] Tan, Zaobao, *et al*. Design and implementation of proxy-based SSO and security audit system for remote desktop access. International Conference on Advanced Intelligence and Awareness Internet (AIAI), 2010.

[7] Bhuyan, Monowar H., D.K. Bhattacharyya, and Jugal K. Kalita. "Network anomaly detection: methods, systems and tools." Communications Surveys & Tutorials, IEEE 16, no. 1 (2014): 303-336.

[8] Rahman, Ahmedur, C. I. Ezeife, and A. K. Aggarwal. Wifi miner: An online apriori-infrequent based wireless intrusion detection system. Knowledge Discovery from Sensor Data, 2012.

[9] Microsoft Corporation. Remote Desktop Protocol (RDP) Features and Performance White Paper.

[10] Real-VNC: An open source software for remote control. http://www.realvnc.com/index.html

[11] Richardon, T. The RFB Protocol. Real-VNC Ltd. http://www-.realvnc.com/docs/rfb proto.pdf, 2009-11-24.

[12] Nan, Z., Paloski, A., & Wang, H. An efficient user verification system via mouse movements." 18th ACM conference on Computer and communications security, 2011.

[13] Muthumari, G., R. Shenbagaraj, & M. Blessa Binolin Pepsi. Mouse gesture based authentication using machine learning algorithm. International Conference on. Advanced Communication Control and Computing, 2014.

[14] RTCA DO-178C, Software Considerations in Airborne Systems and Equipment Certification, dated December 13, 2011

[15] Wang, Ke, and Salvatore J. Stolfo. "Anomalous payload-based network intrusion detection." International Workshop on Recent Advances in Intrusion Detection. Springer Berlin Heidelberg, 2004.

[16] Andersson, Stig, et al. "A framework for detecting network-based code injection attacks targeting Windows and UNIX." 21st Annual Computer Security Applications Conference (ACSAC'05). IEEE, 2005.