A messaging-based system for remote server administration

Marco Ramilli
University of Bologna
Via Venezia 52, 47521 Cesena - Italy
marco.ramilli@unibo.it

Marco Prandini
University of Bologna
Viale Risorgimento 2, 40136 Bologna - Italy
marco.prandini@unibo.it

***Abstract*—The most common method of system administration is accessing the remote system through the network by means of some client-server protocol, giving access to a privileged service always listening on the target system. There are important security and flexibility limitations deriving from the usage of a predictable access port for such a critical application, which can be summarized as the impossibility of attaining a satisfactory trade-off between attack rejection capability and service availability. This work illustrates an alternative solution based on the presence of an intermediate system, acting as a meeting place in between the remote server and its administrator. The resulting architecture eliminates the predictable management port on the server, enhances the availability of the management service by exploiting widespread communication platforms that are likely to be accessible from anywhere, and exhibits a modular structure enabling promising future extensions aimed at overcoming many other issues of the current administration techniques.**

***Keywords*-system administration; instant messaging; botnets;**

I. Introduction

Remote administration is a necessity for the vast majority of Internet servers. Usually, a service runs on the target system, either providing the administrator with a remote view of the locally available administration tools (e.g: remote terminal, remote desktop), or implementing a back-end for the execution of complex commands received through a corresponding front-end (e.g.: web-based administration interfaces).

The access point to the administration service is the obvious target of attacks such as DoS or brute-force authentication attempts. Limiting the impact of these attacks is very difficult; usually one or more reactive or proactive techniques are exploited, as briefly summarized hereinafter.

- Access limitation solutions, like for example account lock-out and connection throttling, react to suspect activity (in terms of failed login attempts or exceedingly high traffic) that could be the symptom of an ongoing attack. In this way, however, they expose the legitimate administrator as well as the attacker to the same risk of being cut out of the server.
- Pre-authentication protocols, as port-knocking [22], [9], [15] and Cryptographically Constantly Changing Port Opening (C3PO) [12], tackle instead the problem of hiding the administration port to everyone but the legitimate administrator; the server recognizes a sequence of specially crafted packets sent by the administrator, and subsequently allows the originating address to access the administration port. The sequence is based on a shared secret, and usually involves sending several packets to the correct TCP or UDP ports within a timeframe. This approach also suffers from various drawbacks: first, temporary lock-out can again be triggered by any malicious or fortuitous event that lets the server receive a wrong sequence; second, a special client is required to execute the protocol; third, traffic directed to truly random, hence unusual ports could be blocked before it reaches the server.
- Host- and network-based intrusion detection systems can help thwarting the attacks before they succeed, but can not guarantee security against the vastly distributed attacks that are presently possible, especially considering the value of the target (that is full control of an Internet host).

However, there is a significant difference between the administration service and the others usually provided by the same host. While the latter must typically be fairly visible to a varied audience, the former is intended solely for the legitimate system administrator. This peculiarity can be leveraged to protect the sensitive administration service from malicious exploitation attempts, by completely changing the access model.

The goal of this research is to devise an unconventional model of communication between the system administrator and the remote administration interface. In the proposed solution, previously outlined in [19], the intrinsic vulnerability of the traditional scheme is addressed by reversing the client-server relation; an administration engine replaces the classical service, originating connections to an intermediate system rather than listening for connections.

The immediate advantage arising from this design choice is that there is nothing to attack on the remote host. On the other hand, the introduction of an additional system in the security chain must be carefully evaluated, to avoid introducing unexpected attack paths that eventually make the system less robust than it originally was. We claim that, if properly modeled and implemented, a platform based on the meeting of the server and its administrator on an intermediate system is expedient in terms of security, availability, usability and opportunity for future extension.

In the following section, we outline the design guidelines for the proposed system and describe the resulting architecture. Then, we proceed to discuss the deriving security issues. Finally, we draw conclusions based both on the present theoretical analysis and on preliminary experimental results.

II. THE PROPOSED SYSTEM

A. Design goals

The devised system pursues four main features that we deem lacking in the traditional remote system administration protocols.

1) Administration port protection. The first line of defense we consider is concealing the very existence of an administration port on the server. The proposed system should replace the visible server access point with an abstract management port (AMP), which is not located in a fixed place, but rather distributed on the network by means of an intermediate communications architecture. We look after a twofold advantage: having no obvious target for attacks on the server, and lowering the motivation of attackers by concealing the AMPs' bound to the corresponding servers (and thus, ultimately, to the value of the attacked system).
2) High availability. In this context, we define availability as the possibility for the legitimate administrator of being able to access the administration interface of a remote server. The majority of the security countermeasures listed in the previous section don't strike a satisfactory trade off between availability and effectiveness. They either risk sacrificing availability in order to block potential attacks, or, where conversely this risk cannot be tolerated, they add little security value. A design goal of the proposed solution, then, is keeping one or more access paths open for the legitimate administrator even under massive suspect activity directed against the administration infrastructure. Defending the administration service against the kind of denial-of-service attack that can tear either the host or the whole surrounding network down, conversely, is not within the reach of the proposed solution.
3) High intrusion prevention capability. As already cited in the previous point, the important quality to pursue is neither availability nor attack rejection alone, but the best trade off between the two features. A design goal of the proposed solution is implementing a flexible communication model between the administrator and the target system. It should be capable of dynamically offering alternative ways of connecting the two ends, when the currently active paths have to be terminated in order to curb an ongoing illicit access attempt.
4) Convenient placement of common functionalities. We foresee to achieve the aforementioned results by interposing an intermediate layer between the administrator and the commonly used administration tools on the server. It makes sense to take advantage of this layer for concentrating there all the functions that are related to the communication between the two sides. Good examples of functions that can be properly placed in the intermediate layer are anomaly/intrusion detection systems and interpreters for customizable, platform-independent administration languages.

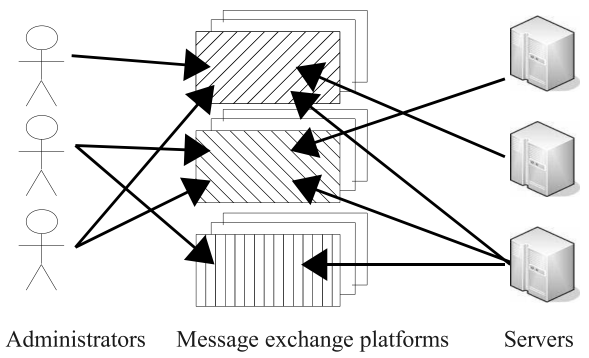

Figure 1. Administrators and servers meeting on message exchange platforms

B. Architecture

To achieve the stated goals, we propose to substitute the remote administration service passively listening for incoming connections with an active system, named RoboAdmin (RA), which connects to a meeting place (MP) where the administrator can contact it, as depicted in Figure 1.

RoboAdmin, as illustrated in Figure 2, exhibits a modular design which favors portable implementations. The role of each component is described hereinafter.

1) Communication Layer: The communication layer provides an abstract functionality of message exchange to RA. It can instantiate one or more communication channels by connecting to the desired MPs, hiding the implementation details of the corresponding platforms to the higher layers. The communication layer lets RA establish a virtual presence on MPs, where the system administrator can find it, and subsequently handles the exchange of messages between the administrator and RA. Existing multi-user communication platforms like the Internet Relay Chat (IRC) , MSN, Skype, etc. are all examples of architectures providing MPs suitable for RA's use; they exhibit all the basic common features needed for our purposes, namely:

- the possibility to create, manage and publish MPs - the place is usually created on a specific host, and can be

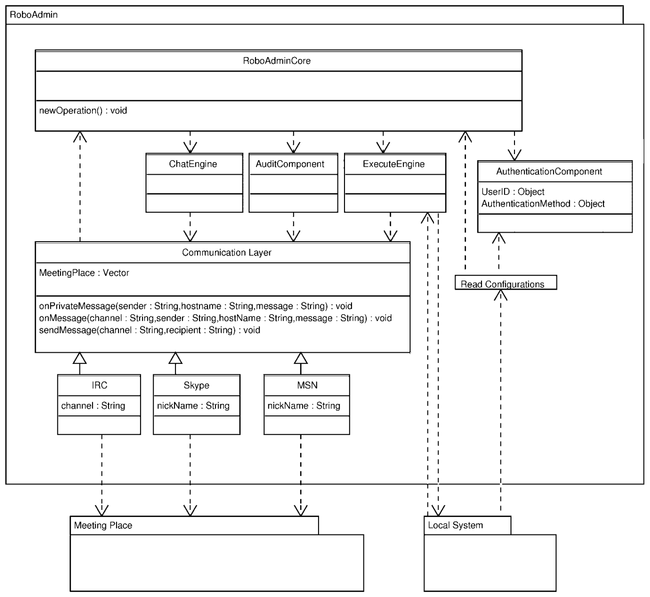

Figure 2. RoboAdmin Architecture

made visible to a global or local network through either a directory or a discovery system;

- the capability to identify users accessing the place by means of a nickname, which can either be independently chosen by them and disambiguated by the system in case of collision, or registered a priori;
- the availability of some authentication and access control mechanism; administrative rights may be reserved to the creator of the place, shared, or delegated;
- the availability of public (broadcast) or private (one-to-one) communication channels.

The adoption of the communication model implemented by this component makes the most of what is needed to reach the design goals. By replacing a listening port with outgoing connections, the most obvious target of attack on the server disappears (clearly, the RA presence on the MP represents just another way of listening to incoming administrative requests; the security consequences of this choice are analyzed in section III). The ability of handling multiple connections at once is the key to achieving the desired satisfactory trade-off between availability and security: closing one of the active connections when an attack against it is detected, even with a rather low confidence, doesn't affect the possibility for the administrator to reach the server by means of the remaining ones. In other words, a very effective intrusion prevention policy can be enacted to tell the real administrator from an attacker trying to access the server through RA, because high false-positive rates of detection (usually associated with low false-negative rates) can be tolerated almost without consequences, as far as it is possible for RA to replace the "compromised" presence on an MP with another one.

Moreover, the choice of exploiting widespread communication platforms benefits the availability of the administration port also when an administrator needs to access his servers in an unforeseen situation, for example from a public Internet access point. It is much more likely he will be able to find an MSN client than an SSH client, let alone a specialized application needed to handle port knocking or cryptographic port choosing (both of which are most probably going to be blocked by a firewall, anyway).

2) Chat Engine: The usage of public communication platforms as MPs has two implications. First, RA should be prepared to handle the dialog with users other than the legitimate administrator without raising too much awareness about its nature. Second, RA must respect the rules of the specific MP to avoid being banned from it. For example, exceedingly long periods of silence could trigger anti-bot rules. In fact, the idea of sending a software agent (bot) on chat rooms or similar communication platforms is certainly not new, even if the reasons are usually malicious [21]; the bots can listen to the conversation between the real users, looking for sensitive data, or they can try to gain administrative rights for the meeting place. Research in the Artificial Intelligence field produced some interesting results in its quest to devise a software able to perform human-like conversation, thus passing the famous Turing test [23]. This goal has been pursued though different approaches, some exploiting clever but not really "intelligent" linguistic tricks [10], others building more complex AI-based system with learning abilities [7]. In the year 1994, Maudin coined the term chatterbot in its paper [16] describing the various conversational programs that appeared at the time. The chat engine, by integrating this kind of conversational technologies, pursues the objective of disguising RA among the other inhabitants of the MP, and irrevocably intercepts any dialog not leading to the start of a real administration session.

3) Authentication Component: When an administrator wants to access a server, the first interaction he attempts with RA is the authentication procedure. The authentication component handles this phase, implementing any kind of suitable authentication backend (e.g. standard passwords, one-time passwords, etc.). Of course, if the administrator is using a standard messaging client, the possibility of participating in strong authentication protocols (e.g. cryptographic challenge-response) will be quite unlikely, but the authentication component can easily accommodate also that kind of feature as well. If the administrator can use some kind of auxiliary device to perform the required computations, by copying the challenge from the messaging client and pasting the response back on it, very secure (if somewhat awkward) authentication procedures can be supported.

4) Executive Component: Once RA has authenticated the administrator, the executive component is put in charge of interfacing him with the system. In the simplest scenario, this could be a transparent wrapper of a standard shell on the system, on one side passing the messages received from the communication layer as command lines, and on the other side intercepting their output and writing it back. The presence of a wrapper of this kind, however, offers other interesting possibilities too. As a simple but effective translator of imperative commands, this component can allow the administrator to use a platform-independent language to execute a wide range of commonplace tasks on heterogeneous target systems. We designed the executive component so that the grammar of the platform-independent language can be customized. Consequently, each administrator can define a unique language that not only is tailored to make the his work easier, but also introduces a further dimension in the search space for the attacker; blind attempts at guessing the correct commands will make the intrusion quickly detectable. As a more visionary development, that leverages the integration of "intelligent" interpreters described above, it would even be possible to foresee the design of a declarative system administration interface, with the ultimate goal of managing the target system in terms of desired configuration goals, leaving to its "intelligence" the task of developing the most effective plan to reach them, as described in [1].

5) Audit and auxiliary Components: Each one of the components hitherto described can report every anomaly or otherwise relevant event to the audit component, which dispatches the notice to the destination configured by the administrator. Clearly, the correct operation of RA's components depends on many configuration options. Among the core functions, a dedicated component deals with their persistence and versioning problems.

C. RA vs. Botnets

As usefully pointed out by one reviewer of a seminal version of this work, two kinds of criticisms could arise when observing the obvious resemblance of RA's architecture to that of a botnet: First, RA could be blocked by the very same countermeasures that are being adopted to curb the spread of botnets. Sadly, the effectiveness of those countermeasures is very limited. Some [18] are based on the analysis of the compromised servers participating in the botnet, either "real" ones or honeypots; some others [26], [18], [2] on the analysis of anomalous network activity generated by those servers. Clearly, neither technique is applicable to servers administrated through RA. Second, the adoption of a botnet-like architecture in a world that is trying to get rid of them is arguably ethical. However, for the reasons outlined on the previous point, RA is not significantly harmed by anti-botnet countermeasures, and consequently it does not need to contribute in any way to the development of anti-countermeasures that could benefit malicious players. Furthermore, RA agents won't try to hamper the MP infrastructure, while this is one of the most annoying behaviors of malicious bots.

Eventually, few but important differences allow us to claim that RA is substantially immune from the theoretical problems that could derive from its botnet-like nature. If these arguments do not convince the reader that RA agents can be accepted on public MPs, it is always possible to foresee the usage of a dedicated infrastructure. Of course this would be a much more complex solution, yet feasible, as demonstrated in a paper [11] published after our first proposal for RA, that estimated the size needed for such an infrastructure to be effective.

III. Security analysis

We are aware that the implicit claim of the presented solution, that is securing a system by making it much more complex, is rather bold. Everybody knows that the chain of security is as strong as the weakest of its links; in this section we analyze all the new links, to prove that, in the worst scenario, the proposed solution is not less secure than the presently adopted ones. The analysis is organized according to the three areas which constitute the RA architecture: the client, the MP and the server. Each of these exhibits peculiar opportunities for a prospective attacker.

A. Client-side Security.

The client platform offers a twofold opportunity for attacks. The choice to adopt widespread, standard client software to communicate with RA means that any attacker can use the same tool as the legitimate administrators. On the one hand, an attacker can install on her own computer a client for some MP technology and try to attack the RA system. On the other hand, an attacker who is in a favorable position can leverage her knowledge about the client to try breaking in the administrator's system or otherwise hijack the the administrator-MP communication. The two scenarios are separately investigated in the following sections.

1) Attacks from the client: Any attacker can run an MP client and try to chat with RA, exactly like she can run one of the common remote access clients and try to connect to a target system. However, while in a client-server model the target is obvious, in the proposed model the attacker must

1) find an MP containing an RA,
2) recognize RA among the other users,
3) attack it in the same way she would try to attack a standard administration service,
4) if successful, operate on the target system so as to avoid the IDS feature integrated in the command interpreter,
5) find out whether the target is interesting or not.

For the sake of a worst-case analysis, we can suppose that items 1, 2, 4 and 5 do not represent hurdles that actually slow down the attacker, but step 3 is at least as complex in RA as in any traditional system. More realistically:

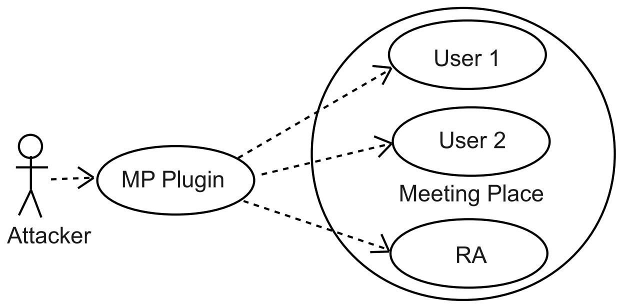

Figure 3. Brute Force Attack From MP Client

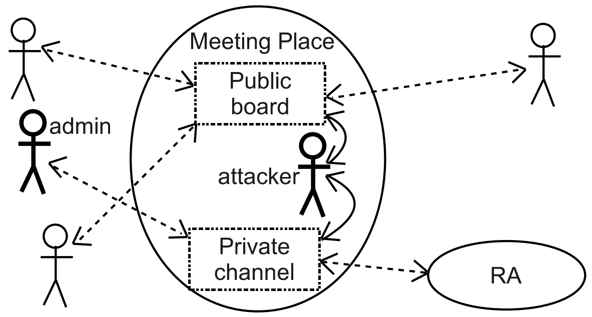

Figure 4. Insider Attack From MP

regarding steps 1 and 2, if RAs are placed on densely populated platforms, the research of an RA will be quite difficult. The five most widespread IM circuits (AIM, Windows Live Messenger, QQ, Skype, ICQ), for example, average a total of about 120 million concurrently online users. Consequently, up to 120,000 RAs could live in those communities and the probability of meeting one by chance would still be lower than one in a thousand (a precise statistical analysis of this scenario will be found in [17]). Moreover, given the size of the research task, even a basic evasive technique enacted by RA can trick the attacker in believing she found yet another regular user;

regarding step 3, RA can adopt quick and effective lockout policies, greatly hindering the attacker's ability to try a brute-force approach, because the legitimate administrator knows alternative MPs where to find other RAs for the same server. The probability that an attacker finds all the instances of RA related to a given server and succeeds in a DoS attack by disabling them all is exponentially decreasing in the number of RA instances. The server can easily enact emergency procedures in this unlikely scenario, for example by sending an RA to a randomly chosen set of altogether new MPs, and telling the administrator their locations by means of an encrypted e-mail or other kind of message;

regarding step 4, with high probability the administrators using RA will enjoy the possibility of customizing the grammar, both to suit their preferences and to trigger the detection of wrong commands issued by an attacker;

regarding step 5, delaying the discovery of the value of a system up to its actual compromise is not of any help against attackers simply looking for the largest number of victims. However, the effective decoupling between a valuable target and its administration port surely helps thwarting the most motivated, and hence dangerous, attacks.

2) Attacks to the client or to the client side of the communication: The RA client is subject to the same attacks as any other software. The attacker can compromise the computer the client runs on, or trick the administrator in installing a trojaned version of the software, or be in a location allowing her to perform a man in the middle attack.

These are not RA-specific problems. The first two issues belong to the domain of system integrity, which is surely taken very seriously by someone using that system as a control center for one or more valuable remote servers. In particular, we expect the administrator to check the signature over any piece of software he installs from the net, even if we acknowledge this is an open cultural problem which many security evangelists are working on [14]. The third issue can be solved by exploiting encryption on the channel between RA and his administrator. Some platforms offer channel encryption as a standard feature, though usually not in a end-to-end fashion, which would allow to solve also the problems described in the following section. On platform that do not offer this option, implementing end-to-end security between the RA client and server is feasible either by modifying the client, or by interposing a security layer between the client and the network (for example see the IRC encryption proxy [20]). This issues are also relevant to the following section; further detail will be given there.

B. Meeting Place security

In the most common implementation scenario for RA, the Meeting Place is the kind of gathering point where standard users chat with each other not caring about privacy and security. Often users don't know who are the owners of the place, nor they realize their powers. Common users do not pose a threat to RA, because MP technologies offer simple ways to establish private channels between the administrator and RA, after they met on the public place. Instead, if we consider the possibility of an attack coming from the MP owner, we must take into account the fact that she is able to perform both passive and active attacks on the communications. It is an Insider Attack problem, as discussed for instance in [4], [5].

Figure 4 shows how this attack scenario is easy to implement for an MP owner. On the left, among users that are talking on MP, an administrator writes the login phrase to RA in a "private" chat in order to start working with the corresponding server. RA acknowledges the authentication and moves itself to the "ready state". The MP owner (attacker, in the center) can both sniff the authentication credentials and hijack an established session, reading sensitive data, learning the administration language, and issuing commands. While the probability of stumbling on a malicious MP owner is in our opinion almost negligible, especially if "renowned" MPs are used, this kind of attack could completely jeopardize RA's security, and thus it must be effectively countered, essentially by guaranteeing that the dialog between the administrator and RA is actually private, either by means of cryptography or bypassing the MP system altogether when exchanging messages. Both of these two approaches have advantages and disadvantages.

The former solution, i.e. exploiting end-to-end cryptography, would be preferable because it allows to leverage any kind of mediated message exchange platform as an MP (with the additional advantage that both the administrator and the server can be hidden behind a NAT device), but the drawback is that a specialized client is needed. For most protocols, however, it is quite easy to implement a client with the necessary modifications. It could be coded as a Java applet, for example, so that an administrator can use it from any web browser, losing the advantage deriving from using standard software, but at least preserving the advantage deriving from using common network protocols, unlikely to be blocked by firewalls.

The latter solution, i.e. bypassing the MP just after the meeting, can be easily implemented on platforms that support moving users among MPs. However, the effectiveness of this maneuver depends on the availability of a trusted MP on which to jump. Since the malicious administrator of the starting MP is almost certainly able to detect the destination of the jump, the trusted MP will quickly become a highly valuable target. It is important to note that the aforementioned countermeasures are appropriate for thwarting the most dangerous, active attacks. However, the fact that a malicious MP owner can easily recognize an RA, and see which server sent it on the MP, remains an open (if not very worrisome) problem for which no straightforward solution seems possible.

Finally, a significant risk is posed by attackers mimicking a RoboAdmin chatterbot on the MP with the intent of stealing the authentication tokens from the administrator. The use of a mutual authentication protocol is advisable, of course taking into account that each phase must be protected against the same attacks described for the one-way authentication scenario. Several works have studied the problem of achieving strong authentication and exchanging cryptographic keys between a client operated by a human player, who has nothing more than his memory to store a secret, and a remote server; see [8] for a good survey, and [25] for SRP, which seems particularly suitable to be integrated in the proposed architecture.

C. *Server side security*

The whole point of the RA architecture is preventing direct attacks to the server. No administration ports are left wide open, with privileged services running behind. The only attacks that are possible on this side of the RA architecture, and which of course could affect also RA's security, are the ones independent from RA itself: exploiting local vulnerabilities through malicious code, exploiting vulnerabilities of remotely-accessible services, exploiting physical vulnerabilities, etc. RA cannot replace the best practices for installing and maintaining a secure system, but does not introduce new means of attacking it, eliminating instead one access path of critical importance.

IV. Implementation and experimental validation

A. *Implementation choices*

The illustrated architecture can be implemented using a wide variety of technologies, supporting almost any combination of target system (i.e. the one we want to remotely administrate) and network infrastructure suitable for building an AMP. In order to build a simple prototype for the experimental validation of the concept, we chose to take advantage of the Internet Relay Chat (IRC) [13], which has been proved successful for a task very similar to ours, albeit with very different intent: controlling infected hosts used for malicious purposes [6]. IRC was probably the first service allowing people to meet on the Internet and exchange text sentences in a many-to-many communications environment, yet still widely used, notwithstanding the large number of alternatives now available. The protocol is well documented and client software is available in many flavors, often even on public-access network stations. Regarding the executive component on the target system, we implemented a configurable command interpreter that wraps the `bash` shell on a Linux machine. An especially challenging part of the implementation was countering control flow injection attacks that could occur in multithreaded distributed systems like RA, if not properly coded: an attacker could be able to block the system by typing multiple requests at the same time. To avoid this vulnerability, the core engine has been implemented according to the well-known concurrent design guidelines by Ben-Ari [3] using Java concurrent libraries.

B. *Experimental results and call for testing*

The prototype has been running on a public host and connected to public IRC servers for more than a year, without incidents so far. The chatterbot actually got contacted by many other users, who rather quickly lost interest in its limited conversational ability, but apparently did not discover its real nature. The interactions have been logged through the audit component; as an example, a "controlled" session is reported in Table I. In this short session all the relevant modes of operation of RA are exemplified. At first, RoboAdmin and the administrator `provaAd` can

```
2008-01-15 10:00:02 RoboAdmin connecting to: localhost
2008-01-15 10:00:32 RoboAdmin connected to: localhost
2008-01-15 10:00:32 RoboAdmin joined to: #null
2008-01-15 10:01:56 provaAd(87.19.242.119) wrote in private chat: Hi, How Are you ?
2008-01-15 10:01:56 RoboAdmin is talking through Eliza Intelligent message( Does that question interest you ? )
2008-01-15 10:02:13 provaAd(87.19.242.119) wrote in private chat: heheh, because I'm quite afraid for you !
2008-01-15 10:02:13 RoboAdmin is talking through Eliza Intelligent message( Don't any other reasons come to mind ? )
2008-01-15 10:02:22 provaAd(87.19.242.119) wrote in private chat: !LOGIN! prova prova
2008-01-15 10:02:22 RoboAdmin has authenticated: prova
2008-01-15 10:02:30 provaAd(87.19.242.119) wrote in private chat: !EXECUTE! ls
2008-01-15 10:02:30 RoboAdmin has EXECUTED: ls
2008-01-15 10:03:03 provaAd(87.19.242.119) wrote in private chat: !LOGOUT!
2008-01-15 10:03:03 RoboAdmin has logged out
2008-01-15 10:03:14 provaAd(87.19.242.119) wrote in private chat: NOW EXAMPLE OF ATTACK
2008-01-15 10:03:14 RoboAdmin is talking through Eliza Intelligent message( Please go on. )
2008-01-15 10:03:31 provaAd(87.19.242.119) wrote in private chat: !LOGIN! BruteForce Pass1
2008-01-15 10:03:31 FAKE LOGIN -> From: provaAd (87.19.242.119). He used username =BruteForce, and password =Pass1
2008-01-15 10:03:31 RoboAdmin is killing in: null the nickName provaAd
2008-01-15 10:03:31 RoboAdmin has added to blackList: 87.19.242.119
```

Table I
LOG FILE FRAGMENT SHOWING BOTH A SUCCESSFUL AND A FAILED LOGIN ATTEMPT

be seen joining the channel. Their dialog proceeds onto a private chat; RoboAdmin won't accept any administrative commands on public places. As long as non-administrative sentences are directed to RA, it replies through the chat component, in this case an implementation of the Eliza conversation engine [24]. For the sake of simplicity, in this prototype the administrative commands have been made very visible, using a syntax like `!ALLCAPS!`; the core component recognizes the login sequence and validates it by means of the authentication component. From this point onwards, up to the point when the logout command is issued, the sequence of sentences from the administrator is sent to the executive component. The last lines illustrate the lock-out policy in action; in this case it is configured so that the first failed login attempt causes the permanent ban of the offending user (identified by means of the nickname and/or the source IP address). The policy will commonly be stricter, since it is reasonable to assume that a real administrator won't waste time chatting with RA: the very first interaction can then be used to flag a user as a potential attacker (or simply an innocent passerby), with the consequence of ignoring any subsequent administrative dialog attempt. The usability tests have been quite successful, showing satisfactory ease-of-use and low performance hit. Regarding security aspects, it is hard to tell how successful RoboAdmin was in attaining its goal of disguising its presence, because the existence of this project is unknown to most of the potential attackers. It would certainly be interesting to gather a community of volunteers to perform realistic tests. For this reason, RoboAdmin is publicly available at `http://roboadmin.sourceforge.net/`.

V. Ongoing work

Currently, modules for platforms other than IRC are being written in order to make the system more widely testable. The most widespread platforms, with many millions of users throughout the world, namely AOL Instant Messenger, ICQ, Skype, Windows Live Messenger and Yahoo! Messenger, all offer APIs for an easy integration within third-party code. On a more structural level, the communication layer is undergoing a redesign procedure to make it work according to dynamic resource discovery paradigms. Presently, RA and the administrator need to agree upon (a set of) MPs in advance, while they could leverage discovery services to dynamically find each other. Among the planned future functionalities, a very interesting one is the possibility of creating private group chat-rooms, where an administrator could manage a whole cluster of servers with a single command. The intrinsic capability of a RoboAdmin agent to send instances to mutually unrelated Meeting Places, and then to analyze and to correlate any activity they observe, could be leveraged to integrate intrusion detection capabilities directly within RA. It is not unreasonable to assume that RA can exhibit a much more stealthy and effective behavior than any attacker trying to discover it by trial and error. If the IDS-like capability proves to be effective enough, it could be evolved into a research goal of its own.

VI. Conclusions

In this paper we presented the architecture and implementation of an unconventional approach to remote system administration. We favored a bottom-up approach in which we started by building a working prototype, that could prove the feasibility of some solutions, and then proceeded in the direction of a necessary theoretical analysis of the usability and security characteristics of the proposed system. The result is a RoboAdmin, a "bot" capable of disguising its presence within any communication infrastructure designed to put users in contact one another, like IRC channels. The infrastructure is suited to let a system administrator contact RA, which then can act as an intermediary between the administrator and any management tool on the target system, possibly hiding platform-specific details during the process. The security of this realization of the Abstract Management Port concept has been analyzed, and found to rely on various factors that add up to the standard problem that any attacker faces of faking the authentication credentials of a legitimate administrator, namely: the difficulty of finding the bot and of associating it to a specific server, and the very effective anti-brute-force measures that can be put in place. The main concerns regard the privacy of sensitive data in unlikely, but dangerous situations that see a motivated attacker as the owner of the meeting place or as a fake RA. Of course it is not possible to use passive authentication protocols within a public channel. The usage of cryptography would solve the problems of secure and mutual authentication, but as a drawback requires not-so-common clients, whereas a very interesting feature of the proposed system would be the possibility to use software commonly found even in public Internet access points. The system has been implemented and tested with real usage activities. The preliminary results are of somewhat limited usefulness, given the fact that no real attacker knows RoboAdmin, yet positive. Moreover, the proposed architecture envisions many possibilities for future improvements of system administration models, especially in the direction of building a proactive and autonomous administration agent.

References

[1] D. J. Adams. Is jabber's chatbot the command line of the future? http://www.openp2p.com/pub/a/p2p/2002/01/11/jabber_bots.html.

[2] S. Antonatos, P. Akritidis, E. P. Markatos, and K. G. Anagnostakis. Defending against hitlist worms using network address space randomization. In *WORM '05: Proceedings of the 2005 ACM workshop on Rapid malcode*, pages 30–40, New York, NY, USA, 2005. ACM.

[3] Ben Ari and M. Ben-Ari. *Principles of Concurrent Programming*. Prentice Hall Professional Technical Reference, 1982.

[4] Matt Bishop. The insider problem revisited. In *NSPW '05: Proceedings of the 2005 workshop on New security paradigms*, pages 75–76, New York, NY, USA, 2005. ACM.

[5] Matt Bishop. Position: "insider" is relative. In *NSPW '05: Proceedings of the 2005 workshop on New security paradigms*, pages 77–78, New York, NY, USA, 2005. ACM.

[6] J. Canavan. The evolution of malicious irc bots, white paper, symantec security response, 2005. http://securityresponse.symantec.com/avcenter/reference/the.evolution.of.malicious.irc.bots.pdf.

[7] Rollo Carpenter. Jabberwacky. http://www.jabberwacky.com/.

[8] Saikat Chakrabarti and Mukesh Singhal. Password-based authentication: Preventing dictionary attacks. *Computer*, 40(6):68–74, 2007.

[9] Rennie deGraaf, John Aycock, and Michael J. Jacobson Jr. Improved port knocking with strong authentication. In *ACSAC*, pages 451–462, 2005.

[10] A. I. Foundation. A.l.i.c.e. http://www.alicebot.org/.

[11] Jérôme François, Radu State, and Olivier Festor. Botnets for scalable management. In Alexander Clemm, Lisandro Zambenedetti Granville, and Rolf Stadler, editors, *DSOM*, volume 4785 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2007.

[12] John Graham-Cumming. Cryptographically constantly changing port opening. http://shimmer.sourceforge.net/.

[13] C. Kalt. Internet relay chat series, rfc2810-13. http://www.rfc-editor.org/.

[14] Nancy G. Leveson. Software safety: why, what, and how. *ACM Comput. Surv.*, 18(2):125–163, 1986.

[15] Antonio Izquierdo Manzanares, Joaquín Torres Márquez, Juan M. Estévez-Tapiador, and Julio César Hernández Castro. Attacks on port knocking authentication mechanism. In *ICCSA (4)*, pages 1292–1300, 2005.

[16] M. L. Mauldin. Chatterbots, tinymuds, and the turing test entering the loebner prize competition. In *Proc. of AAAI-94*, pages 16–21, Seattle, WA, 1994.

[17] Marco Prandini and Marco Ramilli. Redesigning remote system administration paradigms for enhanced security and flexibility. *Computer Standards & Interfaces*. Submitted for pubblication.

[18] Moheeb Abu Rajab, Jay Zarfoss, Fabian Monrose, and Andreas Terzis. A multifaceted approach to understanding the botnet phenomenon. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 41–52, New York, NY, USA, 2006. ACM.

[19] Marco Ramilli and Marco Prandini. Roboadmin: A different approach to remote system administration. In *Proc. 5th International Workshop on Security in Information Systems, WOSIS 2007, Funchal, Madeira, Portugal, June 2007*, pages 43–52. INSTICC Press, 2007.

[20] Salvatore Sanfilippo. Encrirc - irc encryption proxy. http://www.hping.org/encrirc/.

[21] Charlie Schluting. Botnets: Who really "0wns" your computers? http://www.enterprisenetworkingplanet.com/netsecur/article.php/3504801.

[22] Lawrence Teo. Port scans and ping sweeps explained. *Linux J.*, page 2.

[23] A. Turing. Computing machinery and intelligence. *Mind*, 49:433–460, 1950.

[24] Joseph Weizenbaum. Eliza - a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 9(1):36–45, 1966.

[25] Thomas D. Wu. The secure remote password protocol. In *NDSS*. The Internet Society, 1998.

[26] Ying Zhang, Evan Cooke, and Z. Morley Mao. Internet-scale malware mitigation: combining intelligence of the control and data plane. In *WORM '06: Proceedings of the 4th ACM workshop on Recurring malcode*, pages 33–40, New York, NY, USA, 2006. ACM.