RoboAdmin

A different approach to remote system administration

Marco Ramilli
marco.ramilli@ieee.org

Marco Prandini
marco.prandini@unibo.it

The 5th International Workshop on Security in Information Systems (WOSIS-2007)
Funchal, Madeira – Portugal
July 12-13, 2007
Current approach (1)

Usually, a system administrator must connect to remote systems by means of a standard client-server protocol to administrate them.

This approach is vulnerable to:

- *Man In The Middle attacks*
- *Information Gathering attacks*
- *Local applications vulnerabilities*
Current approach (2)

Problems arise from visibility issues as well as technical issues

- An attacker can deduce through the remote service port (for example ssh: 22 tcp) that a particular system (of known value) is administered by a foreign user (likely, with unlimited rights)
- The attacker focuses on compromising this attractive target using every technique in his possession.
Proposed approach

Ideally

- granting easy access to the legitimate administrator
  - consistent interface
  - maximize availability

- preventing infrastructural information gathering by attackers
  - deter enumeration and brute force attacks

- hiding the correlation between the administration channel and the target system
Abstraction

- The abstract model for the proposed solution will be called RoboAdmin and it will be an Intelligent, distributed, easy to understand Finite State Automata (FSA) situated in a public Meeting Place (MP).
- With the aim of disguising its nature, the public place is not set up with the specific task of receiving RoboAdmins, but instead a “real” meeting place is chosen, where RA is surrounded by human people.
Proposed Model (1)
Information Gathering

Stealth Scan.

Is this server administrated from remote?
Proposed Model (2)

- Every Server sends a RoboAdmin to one or more Meeting Places where it “melts in the crowd”
- Every System administrator who wants to administrate his system enter the MP and asks RoboAdmin to execute some task

⇒ RoboAdmin has to distinguish a real, authorized administrator from a user simply willing to chat
Every RoboAdmin has four main components:

- **Communication layer**: allows communication between RoboAdmin and MP
- **Chatter bot**: allows RoboAdmin to chat with regular users
- **Authentication module**: allows RoboAdmin to securely distinguish the administrator from the casual user
- **Command Interpreter**: allows RoboAdmin to understand the administrator’s orders

When the administrator successfully authenticates with RoboAdmin, the system builds a point-to-point communication channel between them.
Behavioral Model

Start

Message

Authentication Request? Yes No

Authentication

Yes No

User simulation

End1

End2

End2

Logout? Yes No

Execute

Authrification Failure? Yes No

Send Output

Right Command? Yes No
Behavioral Model

- **Start**
  - **Message**
  - Authentication Request? Yes
  - No
    - **User simulation**
    - **End1**
  - Yes
    - **Authentication**
    - Authentication Request? Yes
      - **End2**
      - Logout? Yes
        - Execute
        - Send Output
      - No
        - Authentication Failure? Yes
          - **End2**
          - Logout? Yes
            - Execute
            - Send Output
          - No
            - **End2**
        - No
          - **End2**
          - Logout? Yes
            - Execute
            - Send Output
          - No
            - **End2**
  - **Command Interpreter**
    - Right Command?
      - Yes
        - **End2**
      - No
        - **End2**

<trigger>[<guard>]/<effect>
Behavioral Model

Start

Message

<trigger>[<guard>]/<effect>

Authentication Request?

Yes: Authentication

No: User simulation

End1

End2

Logout?

Yes: Execute

No: Send Output

Authentication Request?

Yes: Authentication

No: Authentication Failure?

Yes: Right Command?

No: End1

End2

Authentication Failure?

Yes: Right Command?

No: End1

End2

Right Command?
An important goal is achieving the ability of RoboAdmin to be placed among humans users, disguising as one of them. Only the administrator knows RoboAdmin’s identity and knows how to contact it. For this purpose our first prototype exhibits a rudimentary intelligent behavior, implemented as a prolog theory.
Two clauses on the TuProlog platform: the first one (d_c) coding “known questions” and the second one (d_n_c) coding “unknown questions”, structured as follows:

\[
\begin{align*}
d_c([<\text{expected_question}>|T], [<\text{related_answer}>|O]) & :- !, \ d_c(T, O). \\
d_n_c([], "<\text{escape_sentence}>") & .
\end{align*}
\]

Where:
- `<expected_question>` represents a string known RA
- `<related_answer>` represents a plausible follow-up to `<expected_question>`
- `<escape_sentence>` represents a plausible formula for terminating the conversation after the reception of an unrecognized sentence []

TuProlog (also called 2P) is a Java-based light-weight Prolog engine developed at the Alma Mater Studiorum - Università di Bologna. TuProlog has been designed to be one of the basic bricks of Internet applications and infrastructures; this purpose has dictated its main characteristics, such as deployability, lightness, dynamic configurability, integration with Java and ease of interoperability.
User Simulation – state diagram
User Authentication

- It is important to understand which user is the administrator so the authenticator module must offer a challenge.
  - In our prototype the challenge is a simple user / password authentication scheme
- After a successful authentication the administrator is able to communicate with RoboAdmin
  - In the current prototype through a simple grammar interpreter
An important RoboAdmin’s feature is that its language is totally configurable.

Currently, the recognized grammar follows the following forms:

```
<TOKEN> ::= “REGISTER” | “IDENTIFICATION” | “IDENTIFY” | “EXIT” |
         | “ABOUT” | “JOIN” | “SERVER&CHANNEL” | “REPEAT” | “EXECUTE”
<SH-C> ::= {<shell_command>}
COMMAND ::= !<TOKEN>! <SH-C>
```
Administrative interaction - example

As a simple prototypal implementation, the interpreter recognizes an escape sequence which triggers the execution of Unix shell commands.

- of course only after an User Authentication
- shows the administrator the results
  - For instance:
    - !EXECUTE! route add -net 10.0.0.8/30 gw 192.168.3.1
    - !EXECUTE! route -n
- simple but very effective for text-based interfaces
The real question is:

- Can RoboAdmin increment the administrative task security with respect to the current client/server paradigm?

With RoboAdmin we introduce 4 security layers:

1. Finding RoboAdmin
2. Meeting Place authentication
3. RoboAdmin identification
4. RoboAdmin language variability
5. RoboAdmin authentication

Layers 2 and 5 provide traditional, possibly quite strong, authentication steps.

Layers 1, 3, and 4 offer a weaker kind of protection (randomization), but nonetheless can prove quite effective at “intrusion detection”
Meeting places are countless and distributed on the Internet

Meeting Places have a login procedure

- For instance: if we consider IRC as MP, an IRC-Rooms can require a username and password login procedure

An attacker who wants to exploit RoboAdmin, after a successful entry into an MP, has to guess who RoboAdmin is

- If we consider IRC as MP, in a purposely selected IRC-Room there will be many users, so as to make it difficult for the attacker to understand which is(are) RoboAdmin instance(s)

- Evasive techniques can be put in place by RA by observing, for example, an unusual pattern of communication indicating that a user is actually an attacker trying to discover RA instances
Security of the model (3)

- An attacker who wants to exploit RoboAdmin has to understand which grammar it recognizes
  - For instance: The attacker must know if the used grammar is: 
    `!EXECUTE! <command>` or maybe `!ESEGUI!<command>`
  - Again, it’s very easy to detect guessing attempts

- RoboAdmin has its own Authentication module which filters the attempts of normal users to become administrator
  - For instance: if we consider IRC as MP, theoretically it is possible that every IRC chat user queries RoboAdmin but only who knows the right authentication procedure and holds the correct tokens is able to become administrator
Prototype (1)

- Current prototype structure
- MP on IRC
- Planned extension to other platforms
Prototype – regular chat example
Prototype – administrative dialogue
Conclusions / Future work

- Alternative approach to establishing an administrative dialogue with a remote system

- Functional
  - Prototype based on a widespread infrastructure, with standard client software
  - Currently maps native shell commands on a text interface, can be generalized to provide an abstraction of the administration language

- Secure
  - Abstract management port, cannot be found on the target system
  - Makes easy distinguishing unauthorized access attempts from legitimate ones → easy to lock out “brute forcers” without inconvenience for the administrator