In *Proc. 15th International Financial Cryptography Conference*, January 2011

Absolute Pwnage: A Short Paper About The Security Risks of Remote Administration Tools

Jay Novak, Jonathan Stribley, Kenneth Meagher, and J. Alex Halderman

The University of Michigan
{novakjs, stribs, meagherk, jhalderm}@umich.edu

Abstract. Many IT departments use remote administration products to configure, monitor, and maintain the systems they manage. These tools can be beneficial in the right hands, but they can also be devastating if attackers exploit them to seize control of machines. As a case study, we analyze the security of a remote administration product called Absolute Manage. We find that the system's communication protocol suffers from serious design flaws and fails to provide adequate integrity, confidentiality, or authentication. Attackers can exploit these vulnerabilities to issue unauthorized commands on client systems and execute arbitrary code with administrator privileges. These blatant vulnerabilities suggest that remote administration tools require increased scrutiny from the security community. We recommend that developers adopt defensive designs that limit the damage attackers can cause if they gain control.

1 Introduction

Remote administration products allow system administrators to manage collections of machines from a central location. These tools carry inherent security risks. If an attacker can exploit them to issue unauthorized commands, he may be able to take control of client machines. The question is, do the designers of remote administration software take adequate steps to protect the security of their users?

As a case study, we analyzed the security of Absolute Manage [1], a remote administration tool by Absolute Software. Absolute Manage has has been deployed by companies, universities, and school districts throughout the U.S. [1]. It has been in the news since February 2010, when a school district in Pennsylvania was alleged to be using it to spy on students at home via their laptop webcams [10]. We selected it for our study after this controversy brought it to our attention. We had no reason to believe its security would be particularly weak or strong.

We began with black-box testing, through which we determined that an attacker with control of the network could subvert local clients and run arbitrary code. Following this initial result, we used the IDA Pro disassembler to understand the software's communication protocol and security mechanisms, which we found to contain significant design flaws. These vulnerabilities allowed us to develop attacks that require less control over the network while still providing complete adversarial control over the client.

The problems we found in Absolute Manage suggest lessons for remote administration products more broadly. We observe that the design flaws we uncovered were elementary mistakes that would not have gone unnoticed in even the most basic security review. Given the magnitude of the risks these products pose when they are vulnerable, we recommend that developers adopt defensive design and programming practices. The goal should be to ensure that, even if an attacker is able to issue unauthorized commands through the software, the damage he can cause will be limited.

Outline Section 2 introduces Absolute Manage and the software's communications protocol. Section 3 describes serious vulnerabilities we found in its encryption and authentication. Section 4 explains ways attackers could exploit these flaws to take control of clients. Section 5 discusses ways to mitigate the problems and draws broader security lessons. We survey related work in Section 6, and we conclude in Section 7.

We present additional details in the extended version of this paper, available at http://www.cse.umich.edu/~jhalderm/.

2 Background

Absolute Manage is a product of Absolute Software, which purchased it from Pole Position Software in December 2009 for $12.1 million and 500,000 shares of common stock [2]. Using server-side tools, administrators can instruct clients to install programs, apply software updates, run scripts, take screenshots, or execute code, among other functions. Clients also report status information to the server at regular intervals (by default, every 15 minutes). The client software supports Windows and Mac OS X.

One place where Absolute Manage is used is Lower Merion School District in eastern Pennsylvania, which installed it on laptops issued to around 1800 high school students. In February 2010, one of those students, Blake J. Robbins, sued the district in federal court, alleging that school officials violated students' privacy rights by secretly using the laptop cameras to photograph them in their homes [10], using the Absolute Manage TheftTrack feature.

The Absolute Manage software suite includes Absolute Manage Agent, which runs on client machines, and Absolute Manage Server. Both sides accept TCP connections; by default, clients listen on port 3970 and servers listen on port 3971. The server can issue commands as a response to the heartbeat or by directly contacting the client. If a command cannot be delivered, it is queued and resent in response to the next heartbeat.

The body of each message is an XML-formatted property list. The properties include:

- *AgentSerial* A unique identifier for the client that is randomly generated on installation.
- *AdminUUID* A unique identifier for the server that is randomly generated on installation.

- *CommandUUID* A unique identifier for the command that is randomly generated when it is issued.
- *CommandID* A number that specifies the command to be executed and determines the format of the *CommandParameters* structure.
- *SeedValue* A 192-bit binary value used to authenticate the server.

Prior to transmission, all messages are compressed with `zlib` and encrypted using the Blowfish cipher [11] operating in ECB mode.

3 Vulnerabilities

Due to a number of design flaws, the Absolute Manage protocol fails to provide adequate integrity, confidentiality, or authentication.

3.1 Defective Encryption

The Absolute Manage developers opted for a simple cryptographic design: all clients and servers use the same hard-coded secret keys every time, for every message. We were able to discover the keys by examining the client program with IDA Pro. There are at least four keys used for different purposes: two are based on a German phrase and differ only in punctuation, one is a minor corruption of a common colloquial expression, and one appears to be a snide remark about a design choice. All can be exposed by running the `strings` command on the client binary.

Using hard-coded secret keys is highly risky, since an attacker who manages to extract them from one copy of the software can then attack all the other copies. In the case of Absolute Manage, an attacker who learned the keys could decrypt intercepted protocol messages, including messages containing sensitive private data or proprietary software; he could act as a man-in-the-middle and arbitrarily modify the contents of messages in transit; or he could generate new encrypted messages from scratch and pass them to servers and clients. Another important flaw is that the protocol uses ECB mode, so blocks are encrypted independently and deterministically. This lets attackers compromise the protocol even without knowing the keys.

3.2 Defective Authentication

Since remote administration software gives parties the ability to control the machine, it is essential to ensure that only *authorized* third parties can do so. Unfortunately, Absolute Manage uses an extremely weak authentication mechanism for its client-server communication.

When the client receives a command message, it tries to confirm that it originated from an authorized server. Each server has a unique SeedValue that it includes in all its command messages, and clients discard commands that do not have the expected value. In contrast, the AdminUUID, AgentSerial, and CommandUUID properties can have arbitrary values.

The SeedValue property is a random-looking 192-bit string, so one might expect it to be difficult to guess. In fact, it carries very little entropy. If we decrypt our test server's SeedValue using Blowfish in ECB mode and a different hard-coded key, we get the following bytes:

```
00 00 00 0E 00 31 00 34 00 30 00 31
00 34 00 37 00 35 00 00 00 00 00 00
```

The first four bytes are a length specifier, and the trailing zeroes are padding. After removing these, we are left with the UTF-16 encoding of a 7 character string: `1401475`. This is the server's "serial number," which was provided by Absolute Software along with the product activation key when we purchased our license. If all server serial numbers are 7 digits like ours, and they are randomly assigned, then the SeedValue property contains about 23 bits of entropy. We suspect the assignment is nonrandom, so the actual entropy may be much less.

Furthermore, in certain situations, clients do not apply any authentication at all. One example is upon client initialization. The client discovers its server's SeedValue by *asking the server*. It sends a heartbeat message with the NeedSeedValue property set to true. Until the client receives a NeedSeedValue response from the server, it defaults to accepting commands with any SeedValue. Clients do not store the SeedValue to disk, so they need to ask the server again every time the software starts.

An additional vulnerability is that the servers do not authenticate messages from clients. Any correctly formatted message sent to the server is processed as a valid message, no matter where it originated or who sent it. Any client can send the server a heartbeat message with NeedSeedValue set to true, and the server will respond with its SeedValue.

4 Attacks

There are a variety of ways that attackers could exploit the vulnerabilities we identified to issue unauthorized commands to Absolute Manage clients, including commands that silently install and run arbitrary code with administrative permissions. We demonstrated several of these attacks in a testbed network and measured their performance; for details, see the extended version of this paper.

4.1 Sniffing Attacks

An attacker who can observe Absolute Manage traffic can use a number of techniques to identify clients to target and to learn the server's SeedValue, with which he can issue arbitrary commands to all the server's clients. The most basic attack is to listen for connections initiated by the server, which can be recognized by the default client port number. The destination IP address identifies a potential victim, and, since server-originated commands always contain the SeedValue, the attacker can decrypt them to learn it.

4.2 Guessing Attacks

Adversaries anywhere on the Internet can use other attacks to target any Absolute Manage client with a publicly accessible IP address. For instance, given the address of a target, the attacker could use a brute-force attack to quickly guess the expected SeedValue.

We experimentally measured the performance of a SeedValue guessing attack in a laboratory environment. Our attacker software, a multithreaded Python program, attempts to impersonate the server and repeatedly sends benign commands, each time embedding a different guess for SeedValue. The Absolute Manage client responds with an acknowledgment message containing the CommandResultError property, which has a nonzero value if the SeedValue guess is wrong. Our program attempts different random guesses until the client indicates a successful guess.

Using the multithreaded approach, we were able to test an average of 330 guesses per second on a fast network connection. The bottleneck appeared to be the client's software and TCP stack. We found there was very little performance increase when the number of threads exceeded 8 or when attacking from multiple machines. Under these conditions, the expected time to successfully guess the seed would be about four hours, if we assume that the server's serial number contains seven or fewer digits.

4.3 Global Attacks

An attacker who wants to cast a wide net can use a different attack to efficiently target all clients at once. The first stage of the attack is to build a dictionary of server SeedValues. Servers running Absolute Manage have a unique signature that can be identified using a port scanner such as `nmap` [5]. With such a tool, the attacker can perform an Internet-wide scan for publicly addressable servers. Whenever he finds a server, the attacker sends it a heartbeat message asking it to return its SeedValue, and he adds the response to his dictionary. In the second stage of the attack, the attacker performs further network scanning to locate clients and uses the SeedValue dictionary to mount rapid guessing attacks against them. Using this attack, an adversary can take control of a large fraction of the publicly addressable machines running the Absolute Manage client.

5 Defenses and Lessons

Absolute manage released a new version of the product in mid 2010 that modifies the protocol and adds an SSL-based transport. While SSL is likely to be a significant improvement over the system's current encryption and authentication methods, there are many things that could still go wrong, particularly with certificate management.

Absolute Manage users running older versions need to take immediate steps to protect themselves from the attacks we have described. For the rest of us, the problems in Absolute Manage carry lessons about security risks in remote administration software more generally and about patterns of security failure.

5.1 Risks of Remote Administration Tools

Remote administration products like Absolute Manage carry large risks because they intentionally create mechanisms that allow remote parties to take control of a machine. There will always be a risk of *abuse* by authorized parties, as alleged in the students' lawsuit against Lower Merion School District, but correctly designed technology should at least prevent unauthorized third-party attacks by making sure only authorized parties can issue commands. This requires getting authentication right—exactly what Absolute Manage failed to do.

Because of these inherent dangers, remote administration software warrants careful security scrutiny during design, implementation, and testing. Furthermore, remote administration software should be designed defensively in order to minimize the harm to users if the authentication does fail. For example, clients could default to allowing only a minimal set of low-risk operations, and enabling additional operations could require physical access by an administrator. Or, if the client was intended to allow software installation but not remote desktop control, it could be designed to only allow the installation of binaries signed with a secondary key controlled by the administrators.

When remote actions do take place, clients should give users prominent notification and even a chance to cancel or postpone the activity. Howell and Schechter [6] recently proposed a UI paradigm for giving users control over sensor data that might be applicable in this context. Keeping users informed about what is happening to their computers would make attacks—and abuse—easier to detect and avoid.

5.2 Hard-Coded Keys as a Vulnerability Pattern

The problems with Absolute Manage are part of pattern of security failures involving hard-coded cryptographic keys. Using hard-coded secrets often negates the benefits of cryptography. It is widely recognized as a recurring vulnerability pattern, and was named one of the CWE/SANS "Top 25 Most Dangerous Programming Errors" for 2010 [4].

Why do developers keep making this mistake? Given that the broader security community treats the use of hard-coded keys with deserved contempt, we might conclude that it is a symptom of broader problems, indicative of companies that are devoid of security culture, process, or training. Yet perhaps this instead reflects a rational choice on the part of developers in light of the "weakest link" nature of security. Suppose you are a small developer with the resources to invest, at most, 50% of the cost of building strong security. Since investing 50% will likely leave the system just as vulnerable as if you had invested 1%, why waste the extra money? Retrofitting security is much harder than building it into a product from the start. If our hypothetical developer later sells the system to a larger company that can afford security in its home-grown products, the purchaser may nevertheless be unable to afford the investment needed to make the system secure.

Under this theory, hard-coded keys are a manifestation of a bimodal phenomenon that causes some rational developers to invest heavily in security and others to essentially give up on it. This suggests that the only way to prevent problems like those in Absolute Manage is to change developers' incentives, either by making security much cheaper for them to build or by increasing the odds that insecurity will harm their profits.

6 Related Work

On Absolute Manage To our knowledge, the first published analysis of Absolute Manage appeared in a blog post by security consultants Aaron Rhodes and stryde.hax [13] in February 2010. Other prior work came after we had finished our investigation but before we disclosed our results to the vendor. In late May, the Threat Level blog reported [15] that researchers from the Leviathan Security Group had discovered the hard-coded keys and demonstrated how they could be used to attack clients on local networks. The researchers have not published the details of their findings, but we infer that their attacks cannot target remote victims.

Other Administration Tools Another remote administration tool by Absolute Software reportedly contains security problems related to authentication. Computrace [1], a theft tracking and recovery tool, uses proprietary code in the system BIOS to resist removal. This support can apparently be exploited by an attacker to make malware harder to detect and remove [9].

Other tools that provide remote desktop control have adopted design philosophies that differ from the approach used in Absolute Manage. Apple Remote Desktop [3] and the Windows Remote Desktop Connection feature [8] normally display prominent notifications that the system is under remote control.

In contrast, Back Orifice [12] is a remote administration program that is designed to hide from users. Due to its potential for malicious use, a number of antimalware programs have added it to their blacklists. Indeed, the distinction between remote administration tools and malware like spyware, back doors, and bots can be a fine line—sometimes, the only difference is who is intended to be in control.

Lest we forget, the Internet's oldest tool for remote administration, `telnet` [7], used no cryptography at all. Incredibly, it took nearly 30 years for a secure alternative [14] to catch on.

7 Conclusions

In this paper, we revealed critical security vulnerabilities in Absolute Manage and demonstrated how attackers could harness them to cause widespread damage. We used these problems as a case study to discuss the broader risks of remote administration software and how they might be mitigated. The blatant vulnerabilities in Absolute Manage suggest that this class of remote administration programs requires greater security scrutiny, particularly in light of the danger such software can pose when commanded by unauthorized third parties. Secure authentication is a necessity, of course, but such products should be further strengthened by employing defensive design and programming techniques.

Acknowledgments

The authors are grateful to Scott Wolchok for his many insights and invaluable technical contributions, and for submitting the paper after the rest of us had fallen asleep. We thank the anonymous reviewers for their constructive feedback, and Jaeyeon Jung for shepherding the work to publication. We also wish to thank Ari Feldman, Ed Felten, Bob Lougheed, Gordon Lyon, Hovav Shacham, Eric Wustrow, and the students of Michigan's winter 2010 Computer and Network Security class for helpful discussions and other assistance.

References

1. Absolute Software. Absolute Manage Web Site. http://www.absolute.com/en_GB/products/absolute-manage.
2. Absolute Software. Absolute Software Acquires LANrev. http://www.absolute.com/company/pressroom/news/2009/12/lanrev, December 3, 2009.
3. Apple. Remote Desktop 3. http://www.apple.com/remotedesktop/.
4. CWE/SANS. 2010 Top 25 Most Dangerous Programming Errors. http://cwe.mitre.org/top25/.
5. Lyon, G. F. *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning.* Insecure, USA, 2009.
6. Jon Howell and Stuart Schechter. What You See is What They Get: Protecting Users from Unwanted Use of Microphones, Cameras, and Other Sensors. *Web 2.0 Security and Privacy,* 2010.
7. Postel, J., Reynolds, J., and Reynolds, J. Telnet protocol specification. STD 8, RFC 854, May 1983.
8. Microsoft. Connect to Another Computer Using Remote Desktop Connection. http://windows.microsoft.com/en-us/windows-vista/Connect-to-another-computer-using-Remote-Desktop-Connection.
9. Alfredo Ortega and Anibal Sacco. Deactivate the Rootkit: Attacks on BIOS Anti-Theft Technologies. *Blackhat,* 2009.
10. Blake J. Robbins and others. Complaint Against Lower Merion School District Et Al. http://docs.justia.com/cases/federal/district-courts/pennsylvania/paedce/2:2010cv00665/347863/1/, February 16, 2010.
11. Bruce Schneier. Description of a new variable-length key, 64-bit block cipher (Blowfish). *Fast Software Encryption,* pages 191–204, 1993.
12. Sir Dystic. Back Orifice. http://www.cultdeadcow.com/tools/bo.html.
13. stryde.hax and Aaron Rhodes. The Spy At Harriton High. http://strydehax.blogspot.com/2010/02/spy-at-harrington-high.html, February 2010.
14. Ylonen, T. SSH–secure login connections over the Internet. *Proceedings of the 6th USENIX Security Symposium,* pages 37–42, 1996.
15. Kim Zetter. School Spy Program Used on Students Contains Hacker-Friendly Security Hole. Threat Level. http://www.wired.com/threatlevel/2010/05/lanrev/, May 2010.