
/THEORY/IN/PRACTICE
Beautiful Security
安全之美（影印版）
一流的安全专家解释他们是如何思考
O'REILLY®
東南大學出版社
Andy Oram & John Viega 编

安全之美(影印版)

Beautiful Security

Andy Oram　John Viega

O'REILLY®

Beijing · Cambridge · Farnham · Köln · Sebastopol · Taipei · Tokyo

东南大学出版社

图书在版编目（CIP）数据

安全之美：英文/（美）奥莱姆（Oram, A.），（美）卫加（Viega, J.）著．—影印本．—南京：东南大学出版社，2010.6

书名原文：Beautiful Security

ISBN 978-7-5641-2271-3

Ⅰ．①安… Ⅱ．①奥… ②卫… Ⅲ．①电子计算机－安全技术 Ⅳ．① TP309

中国版本图书馆 CIP 数据核字（2010）第 089209 号

江苏省版权局著作权合同登记

图字：10-2010-153 号

安全之美（影印版）

出版发行：东南大学出版社
地　　址：南京四牌楼 2 号
出 版 人：江　汉
网　　址：http://press.seu.edu.cn
电子邮件：press@seu.edu.cn
印　　刷：扬中市印刷有限公司
开　　本：787 毫米 × 980 毫米　16 开本
印　　张：18.75 印张
字　　数：464 千字
版　　次：2010 年 6 月第 1 版
印　　次：2010 年 6 月第 1 次印刷
书　　号：ISBN 978-7-5641-2271-3
印　　数：1~1800 册
定　　价：52.00 元（册）

本社图书若有印装质量问题，请直接与读者服务部联系。电话（传真）：025-83792328

Preface

IF ONE BELIEVES THAT NEWS HEADLINES REVEAL TRENDS, THESE ARE INTERESTING times for computer security buffs. As *Beautiful Security* went to press, I read that a piece of software capable of turning on microphones and cameras and stealing data has been discovered on more than 1,200 computers in 103 countries, particularly in embassies and other sensitive government sites. On another front, a court upheld the right of U.S. investigators to look at phone and Internet records without a warrant (so long as one end of the conversation is outside the U.S.). And this week's routine vulnerabilities include a buffer overflow in Adobe Acrobat and Adobe Reader—with known current exploits—that lets attackers execute arbitrary code on your system using your privileges after you open their PDF.

Headlines are actually not good indicators of trends, because in the long run history is driven by subtle evolutionary changes noticed only by a few—such as the leading security experts who contributed to this book. The current directions taken by security threats as well as responses can be discovered in these pages.

All the alarming news items I mentioned in the first paragraph are just business as usual in the security field. Yes, they are part of trends that should worry all of us, but we also need to look at newer and less dramatic vulnerabilities. The contributors to this book have, for decades, been on the forefront of discovering weaknesses in our working habits and suggesting unconventional ways to deal with them.

Why Security Is Beautiful

I asked security expert John Viega to help find the authors for this book out of frustration concerning the way ordinary computer users view security. Apart from the lurid descriptions of break-ins and thefts they read about in the press, average folks think of security as boring.

Security, to many, is represented by nagging reminders from system administrators to create backup folders, and by seemingly endless dialog boxes demanding passwords before a web page is displayed. Office workers roll their eyes and curse as they read the password off the notepad next to their desk (lying on top of the budget printout that an office administrator told them should be in a locked drawer). If this is security, who would want to make a career of it? Or buy a book from O'Reilly about it? Or think about it for more than 30 seconds at a time?

To people tasked with creating secure systems, the effort seems hopeless. Nobody at their site cooperates with their procedures, and the business managers refuse to allocate more than a pittance to security. Jaded from the endless instances of zero-day exploits and unpatched vulnerabilities in the tools and languages they have to work with, programmers and system administrators become lax.

This is why books on security sell poorly (although in the last year or two, sales have picked up a bit). Books on hacking into systems sell much better than books about how to protect systems, a trend that really scares me.

Well, this book should change that. It will show that security is about the most exciting career you can have. It is not tedious, not bureaucratic, and not constraining. In fact, it exercises the imagination like nothing else in technology.

Most of the programming books I've edited over the years offer a chapter on security. These chapters are certainly useful, because they allow the author to teach some general principles along with good habits, but I've been bothered by the convention because it draws a line around the topic of security. It feeds the all-too-common view of security as an add-on and an afterthought. *Beautiful Security* demolishes that conceit.

John chose for this book a range of authors who have demonstrated insight over and over in the field and who had something new to say. Some have designed systems that thousands rely on; some have taken high-level jobs in major corporations; some have testified on and worked for government bodies. All of them are looking for the problems and solutions that the rest of us know nothing about—but will be talking about a lot a few years from now.

The authors show that effective security keeps you on your toes all the time. It breaks across boundaries in technology, in cognition, and in organizational structures. The black hats in security succeed by exquisitely exercising creativity; therefore, those defending against them must do the same.

With the world's infosecurity resting on their shoulders, the authors could be chastised for taking time off to write these chapters. And indeed, many of them experienced stress trying to balance their demanding careers with the work on this book. But the time spent was worth it, because this book can advance their larger goals. If more people become intrigued with the field of security, resolve to investigate it further, and give their attention and their support to people trying to carry out organizational change in the interest of better protection, the book will have been well worth the effort.

On March 19, 2009, the Senate Committee on Commerce, Science, and Transportation held a hearing on the dearth of experts in information technology and how that hurts the country's cybersecurity. There's an urgent need to interest students and professionals in security issues; this book represents a step toward that goal.

Audience for This Book

This book is meant for people interested in computer technology who want to experience a bit of life at the cutting edge. The audience includes students exploring career possibilities, people with a bit of programming background, and those who have a modest to advanced understanding of computing.

The authors explain technology at a level where a relatively novice reader can get a sense of the workings of attacks and defenses. The expert reader can enjoy the discussions even more, as they will lend depth to his or her knowledge of security tenets and provide guidance for further research.

Donation

The authors are donating the royalties from this book to the Internet Engineering Task Force (IETF), an organization critical to the development of the Internet and a fascinating model of enlightened, self-organized governance. The Internet would not be imaginable without the scientific debates, supple standard-making, and wise compromises made by dedicated members of the IETF, described on their web page as a "large open international community of network designers, operators, vendors, and researchers." O'Reilly will send royalties to the Internet Society (ISOC), the longtime source of funding and organizational support for the IETF.

Organization of the Material

The chapters in this book are not ordered along any particular scheme, but have been arranged to provide an engaging reading experience that unfolds new perspectives in hopefully surprising ways. Chapters that deal with similar themes, however, are grouped together.

Chapter 1, *Psychological Security Traps*, by Peiter "Mudge" Zatko

Chapter 2, *Wireless Networking: Fertile Ground for Social Engineering*, by Jim Stickley

Chapter 3, *Beautiful Security Metrics*, by Elizabeth A. Nichols

Chapter 4, *The Underground Economy of Security Breaches*, by Chenxi Wang

Chapter 5, *Beautiful Trade: Rethinking E-Commerce Security*, by Ed Bellis

Chapter 6, *Securing Online Advertising: Rustlers and Sheriffs in the New Wild West*, by Benjamin Edelman

Chapter 7, *The Evolution of PGP's Web of Trust*, by Phil Zimmermann and Jon Callas

Chapter 8, *Open Source Honeyclient: Proactive Detection of Client-Side Exploits*, by Kathy Wang

Chapter 9, *Tomorrow's Security Cogs and Levers*, by Mark Curphey

Chapter 10, *Security by Design*, by John McManus

Chapter 11, *Forcing Firms to Focus: Is Secure Software in Your Future?*, by James Routh

Chapter 12, *Oh No, Here Come the Infosecurity Lawyers!*, by Randy V. Sabett

Chapter 13, *Beautiful Log Handling*, by Anton Chuvakin

Chapter 14, *Incident Detection: Finding the Other 68%*, by Grant Geyer and Brian Dunphy

Chapter 15, *Doing Real Work Without Real Data*, by Peter Wayner

Chapter 16, *Casting Spells: PC Security Theater*, by Michael Wood and Fernando Francisco

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic
: Indicates new terms, URLs, filenames, and Unix utilities.

`Constant width`
: Indicates the contents of computer files and generally anything found in programs.

Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a

significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Beautiful Security*, edited by Andy Oram and John Viega. Copyright 2009 O'Reilly Media, Inc., 978-0-596-52748-8."

If you feel your use of code examples falls outside fair use or the permission given here, feel free to contact us at *permissions@oreilly.com*.

Safari® Books Online

When you see a Safari® Books Online icon on the cover of your favorite technology book, that means the book is available online through the O'Reilly Network Safari Bookshelf.

Safari offers a solution that's better than e-books. It's a virtual library that lets you easily search thousands of top tech books, cut and paste code samples, download chapters, and find quick answers when you need the most accurate, current information. Try it for free at *http://my.safaribooksonline.com/*.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at:

http://www.oreilly.com/catalog/9780596527488

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about our books, conferences, Resource Centers, and the O'Reilly Network, see our website at:

http://www.oreilly.com

CONTENTS

	PREFACE	xi
1	PSYCHOLOGICAL SECURITY TRAPS *by Peiter "Mudge" Zatko*	1
	Learned Helplessness and Naïveté	2
	Confirmation Traps	10
	Functional Fixation	14
	Summary	20
2	WIRELESS NETWORKING: FERTILE GROUND FOR SOCIAL ENGINEERING *by Jim Stickley*	21
	Easy Money	22
	Wireless Gone Wild	28
	Still, Wireless Is the Future	31
3	BEAUTIFUL SECURITY METRICS *by Elizabeth A. Nichols*	33
	Security Metrics by Analogy: Health	34
	Security Metrics by Example	38
	Summary	60
4	THE UNDERGROUND ECONOMY OF SECURITY BREACHES *by Chenxi Wang*	63
	The Makeup and Infrastructure of the Cyber Underground	64
	The Payoff	66
	How Can We Combat This Growing Underground Economy?	71
	Summary	72
5	BEAUTIFUL TRADE: RETHINKING E-COMMERCE SECURITY *by Ed Bellis*	73
	Deconstructing Commerce	74
	Weak Amelioration Attempts	76
	E-Commerce Redone: A New Security Model	83
	The New Model	86
6	SECURING ONLINE ADVERTISING: RUSTLERS AND SHERIFFS IN THE NEW WILD WEST *by Benjamin Edelman*	89
	Attacks on Users	89
	Advertisers As Victims	98

Creating Accountability in Online Advertising 105

7 THE EVOLUTION OF PGP'S WEB OF TRUST 107
by Phil Zimmermann and Jon Callas

PGP and OpenPGP 108
Trust, Validity, and Authority 108
PGP and Crypto History 116
Enhancements to the Original Web of Trust Model 120
Interesting Areas for Further Research 128
References 129

8 OPEN SOURCE HONEYCLIENT: PROACTIVE DETECTION OF CLIENT-SIDE EXPLOITS 131
by Kathy Wang

Enter Honeyclients 133
Introducing the World's First Open Source Honeyclient 133
Second-Generation Honeyclients 135
Honeyclient Operational Results 139
Analysis of Exploits 141
Limitations of the Current Honeyclient Implementation 143
Related Work 144
The Future of Honeyclients 146

9 TOMORROW'S SECURITY COGS AND LEVERS 147
by Mark Curphey

Cloud Computing and Web Services: The Single Machine Is Here 150
Connecting People, Process, and Technology: The Potential for Business Process Management 154
Social Networking: When People Start Communicating, Big Things Change 158
Information Security Economics: Supercrunching and the New Rules of the Grid 162
Platforms of the Long-Tail Variety: Why the Future Will Be Different for Us All 165
Conclusion 168
Acknowledgments 169

10 SECURITY BY DESIGN 171
by John McManus

Metrics with No Meaning 172
Time to Market or Time to Quality? 174
How a Disciplined System Development Lifecycle Can Help 178
Conclusion: Beautiful Security Is an Attribute of Beautiful Systems 181

11 FORCING FIRMS TO FOCUS: IS SECURE SOFTWARE IN YOUR FUTURE? 183
by Jim Routh

Implicit Requirements Can Still Be Powerful 184
How One Firm Came to Demand Secure Software 185
Enforcing Security in Off-the-Shelf Software 190
Analysis: How to Make the World's Software More Secure 193

12 OH NO, HERE COME THE INFOSECURITY LAWYERS! 199
by Randy V. Sabett

Culture 200
Balance 202
Communication 207
Doing the Right Thing 211

13 BEAUTIFUL LOG HANDLING 213
by Anton Chuvakin

Logs in Security Laws and Standards 213
Focus on Logs 214
When Logs Are Invaluable 215
Challenges with Logs 216
Case Study: Behind a Trashed Server 218
Future Logging 221
Conclusions 223

14 INCIDENT DETECTION: FINDING THE OTHER 68% 225
by Grant Geyer and Brian Dunphy

A Common Starting Point 226
Improving Detection with Context 228
Improving Perspective with Host Logging 232
Summary 237

15 DOING REAL WORK WITHOUT REAL DATA 239
by Peter Wayner

How Data Translucency Works 240
A Real-Life Example 243
Personal Data Stored As a Convenience 244
Trade-offs 244
Going Deeper 245
References 246

16 CASTING SPELLS: PC SECURITY THEATER 247
by Michael Wood and Fernando Francisco

Growing Attacks, Defenses in Retreat 248
The Illusion Revealed 252
Better Practices for Desktop Security 257
Conclusion 258

CONTRIBUTORS 259

INDEX 269

CHAPTER ONE

Psychological Security Traps

Peiter "Mudge" Zatko

DURING MY CAREER OF ATTACKING SOFTWARE AND THE FACILITIES THEY POWER, many colleagues have remarked that I have a somewhat nonstandard approach. I tended to be surprised to hear this, as the approach seemed logical and straightforward to me. In contrast, I felt that academic approaches were too abstract to realize wide success in real-world applications. These more conventional disciplines were taking an almost completely random tack with no focus or, on the opposite end of the spectrum, spending hundreds of hours reverse-engineering and tracing applications to (hopefully) discover their vulnerabilities before they were exploited out in the field.

Now, please do not take this the wrong way. I'm not condemning the aforementioned techniques. In fact I agree they are critical tools in the art of vulnerability discovery and exploitation. However, I believe in applying some shortcuts and alternative views to envelope, enhance, and—sometimes—bypass these approaches.

In this chapter I'll talk about some of these alternative views and how they can help us get inside the mind of the developer whose code or system we engage as security professionals.

Why might you want to get inside the mind of the developer? There are many reasons, but for this chapter we will focus on various constraints that are imposed on the creation of code and the people who write it. These issues often result in suboptimal systems from the security viewpoint, and by understanding some of the environmental, psychological, and philosophical frameworks in which the coding is done, we can shine a spotlight on which areas of a system

are more likely to contain vulnerabilities that attackers can exploit. Where appropriate, I'll share anecdotes to provide examples of the mindset issue at hand.

My focus for the past several years has been on large-scale environments such as major corporations, government agencies and their various enclaves, and even nation states. While many of the elements are applicable to smaller environments, and even to individuals, I like to show the issues in larger terms to offer a broader social picture. Of course, painting with such a broad brush requires generalizations, and you may be able to find instances that contradict the examples. I won't cite counterexamples, given the short space allotted to the chapter.

The goal here is not to highlight particular technologies, but rather to talk about some environmental and psychological situations that caused weak security to come into being. It is important to consider the external influences and restrictions placed on the implementers of a technology, in order to best understand where weaknesses will logically be introduced. While this is an enjoyable mental game to play on the offensive side of the coin, it takes on new dimensions when the defenders also play the game and a) prevent errors that would otherwise lead to attacks or b) use these same techniques to game the attackers and how they operate. At this point, the security game becomes what I consider *beautiful.*

The mindsets I'll cover fall into the categories of learned helplessness and naïveté, confirmation traps, and functional fixation. This is not an exhaustive list of influencing factors in security design and implementation, but a starting point to encourage further awareness of the potential security dangers in systems that you create or depend on.

Learned Helplessness and Naïveté

Sociologists and psychologists have discovered a phenomenon in both humans and other animals that they call *learned helplessness.* It springs from repeated frustration when trying to achieve one's goals or rescue oneself from a bad situation. Ultimately, the animal subjected to this extremely destructive treatment stops trying. Even when chances to do well or escape come along, the animal remains passive and fails to take advantage of them.

To illustrate that even sophisticated and rational software engineers are subject to this debilitating flaw, I'll use an example where poor security can be traced back to the roots of backward compatibility.

Backward compatibility is a perennial problem for existing technology deployments. New technologies are discovered and need to be deployed that are incompatible with, or at the very least substantially different from, existing solutions.

At each point in a system's evolution, vendors need to determine whether they will forcibly end-of-life the existing solutions, provide a migration path, or devise a way to allow both the legacy and modern solutions to interact in perpetuity. All of these decisions have numerous ramifications from both business and technology perspectives. But the decision is usually

driven by business desires and comes down as a decree to the developers and engineers.* When this happens, the people responsible for creating the actual implementation will have the impression that the decision has already been made and that they just have to live with it. No further reevaluation or double guessing need take place.

Imagine that the decision was made to maintain compatibility with the legacy technology in its replacement. Management further decrees that no further development or support work will take place on the legacy solution, in order to encourage existing customers to migrate to the replacement.

Although such decisions place burdens on the development in many ways—with security implications—they are particularly interesting when one solution, usually the new technology, is more secure than the other. In fact, new technologies are often developed *explicitly* to meet the need for greater security—and yet the old technology must still be supported. What security problems arise in such situations?

There are different ways to achieve backward compatibility, some more secure than others. But once the developers understand that the older, less secure technology is allowed to live on, solutions that would ease the risk are often not considered at all. The focus is placed on the new technology, and the legacy technology is glued into it (or vice versa) with minimal attention to the legacy's effects. After all, the team that is implementing the new technology usually didn't develop the legacy code and the goal is to ultimately supplant the legacy solution anyway—right?

The most direct solution is to compromise the robustness and security strength of the new technology to match that of the legacy solution, in essence allowing both the modern and legacy technology to be active simultaneously. Learned helplessness enters when developers can't imagine that anything could be done—or worse, even should be done—to mitigate the vulnerabilities of the legacy code. The legacy code was forced on them, it is not perceived to be their bailiwick (even if it impacts the security of the new technology by reducing it to the level of the old), and they feel they are powerless to do anything about it anyway due to corporate decree.

A Real-Life Example: How Microsoft Enabled L0phtCrack

Years ago, to help system administrators uncover vulnerabilities, I wrote a password-cracking tool that recovered Microsoft user passwords. It was called L0phtCrack at the time, later to be renamed LC5, and then discontinued by Symantec (who had acquired the rights to it) due to concerns that it could be considered a munition under the International Tariff on Arms Regulations (ITAR).† Many articles on the Net and passages in technical books have been written about how L0phtCrack worked, but none have focused on *why* it worked in the first

* Or at least it often appears to the developers and engineers that this is the case.

† This might not be the end of L0phtCrack....

place. What were some of the potential influences that contributed to the vulnerabilities that L0phtCrack took advantage of in Microsoft Windows?

In fact, the tool directly exploited numerous problems in the implementation and use of cryptographic routines in Windows. All these problems originated in the legacy LAN Manager (or LANMAN) hash function that continued to be used in versions of Windows up to Vista. Its hash representation, although based on the already aging Data Encryption Standard (DES), contained no salt. In addition, passwords in LANMAN were case-insensitive. The function broke the 14-character or shorter password into two 7-byte values that were each encrypted against the same key and then concatenated. As I described in a post to BugTraq in the late 1990s, the basic encryption sequence, illustrated in Figure 1-1, is:

1. If the password is less than 14 characters, pad it with nulls to fill out the allocated 14-character space set aside for the password. If the password is greater than 14 characters, in contrast, it is truncated.
2. Convert the 14-character password to all uppercase and split it into two 7-character halves. It should be noted that if the original password was 7 or fewer characters, the second half will always be 7 nulls.
3. Convert each 7-byte half to an 8-byte parity DES key.
4. DES encrypt a known constant ("KGS!@#$%") using each of the previously mentioned keys.
5. Concatenate the two outputs to form the LM_HASH representation.

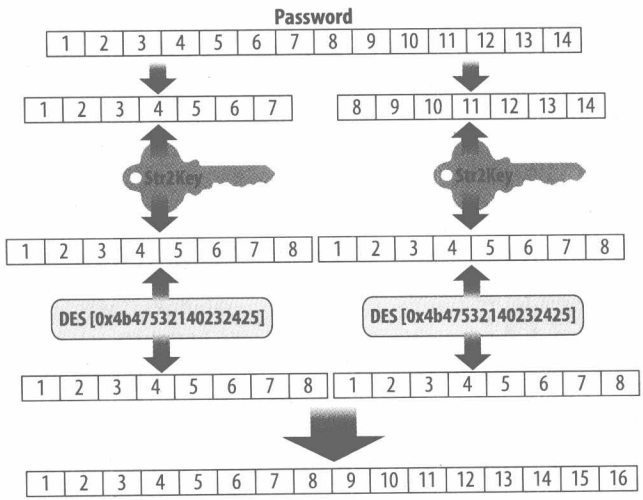

FIGURE 1-1. Summary of old LANMAN hash algorithm

This combination of choices was problematic for many technical reasons.

The developers of Windows NT were conscious of the weaknesses in the LANMAN hash and used a stronger algorithm for its storage of password credentials, referred to as the NT hash. It maintained the case of the characters, allowed passwords longer than 14 characters, and used the more modern MD4 message digest to produce its 16-byte hash.

Unfortunately, Windows systems continued to store the weaker version of each password next to the stronger one—and to send both versions over the network each time a user logged in. Across the network, both the weaker 16-byte LANMAN hash and the stronger 16-byte NT hash underwent the following process, which is represented in Figure 1-2:

1. Pad the hash with nulls to 21 bytes.
2. Break the 21-byte result into three 7-byte subcomponents.
3. Convert each 7-byte subcomponent to 8-byte parity DES keys.
4. Encrypt an 8-byte challenge, which was visibly sent across the network, using the previously mentioned DES keys.
5. Concatenate the three 8-byte outputs from step 4 to make a 24-byte representation that would be sent over the network.

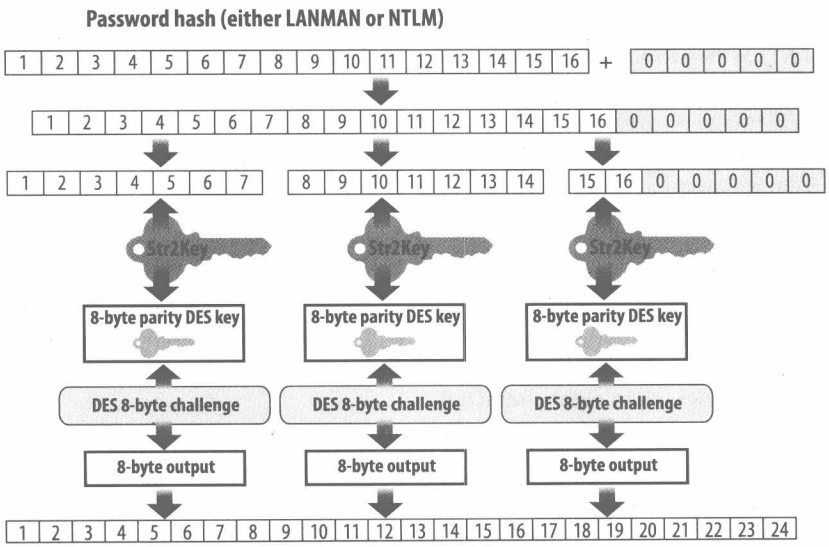

FIGURE 1-2. Handling both LANMAN and NT hashes over the network

Microsoft preferred for all their customers to upgrade to newer versions of Windows, of course, but did not dare to cut off customers using older versions or even retrofit them with the new hash function. Because the password was a key part of networking, they had to assume that, for the foreseeable future, old systems with no understanding of the new hash function would continue to connect to systems fitted out with the more secure hash.

If systems on both sides of the login were new systems with new hash functions, they could perform the actual authentication using the stronger NT hash. But a representation of the older and more vulnerable LANMAN hash was sent right alongside its stronger sibling.

By taking the path of least resistance to backward compatibility and ignoring the ramifications, Microsoft completely undermined the technical advances of its newer security technology.

L0phtCrack took advantage of the weak LANMAN password encoding and leveraged the results against the stronger NTLM representation that was stored next to it. Even if a user chose a password longer than 14 characters, the cracking of the LANMAN hash would still provide the first 14, leaving only a short remnant to guess through inference or brute force. Unlike LANMAN, the NT hash was case-sensitive. But once the weak version was broken, the case specifics of the password in the NT hash could be derived in a maximum of 2^x attempts (where x is the length of the password string) because there were at most two choices (uppercase or lowercase) for each character. Keep in mind that x was less than or equal to 14 and thus trivial to test exhaustively.

Although NTLM network authentication introduced a challenge that was supposed to act as a salt mechanism, the output still contained too much information that an attacker could see and take advantage of. Only two bytes from the original 16-byte hash made it into the third 7-byte component; the rest was known to be nulls. Similarly, only one byte—the eighth—made it from the first half of the hash into the second 7-byte component.

Think of what would happen if the original password were seven characters or less (a very likely choice for casual users). In the LANMAN hash, the second group of 7 input bytes would be all nulls, so the output hash bytes 9 through 16 would always be the same value. And this is further propagated through the NTLM algorithm. At the very least, it takes little effort to determine whether the last 8 bytes of a 24-byte NTLM authentication response were from a password that was less than eight characters.

In short, the problems of the new modern security solution sprang from the weaker LANMAN password of the legacy system and thus reduced the entire security profile to its lowest common denominator. It wasn't until much later, and after much negative security publicity, that Microsoft introduced the capability of sending only one hash or the other, and not both by default—and even later that they stopped storing both LANMAN and NT hashes in proximity to each other on local systems.