Musings About Information Security

Gary C. Kessler
Gary Kessler Associates
Norwich University
Edith Cowan University

An edited, shorter version of this paper appears with the title "Information Security: New Threats or Familiar Problems?" in* IEEE Computer Magazine*, February, 2012, 45(2), 59-65.

What has been will be again, what has been done will be done again; there is nothing new under the sun.

-- Ecclesiastes 1:9, circa 200 B.C.

No it isn't strange, after changes upon changes, we are more or less the same;
After changes, we are more or less the same.

-- Paul Simon, "The Boxer," 1968 A.D.

Abstract

Information security vulnerabilities and threats are alive and well today, but this is nothing new. Attacks on data and telecommunications infrastructures have been going on since the development of those infrastructures, and the attacks adapt to the changing technology. This paper is a personal reminisce of how the attack vectors have changed during the career of the author, but the types of attacks remain eerily similar.

Keywords: information security, history

Introduction

In early 2010, I was asked to give a talk about future trends in information security. As I was putting information together for that talk, I thought about my nearly 40 years in the computer industry and the security issues that I have been dealing with since the 1970s. What I realized is that while many of today's security attack vectors are new, the vast majority of the security threats we have today -- and will have in the future -- have roots and an analog in problems we had in the early computer days. In fact, many of the threats are rooted in the pre-computer, even pre-technology, era (Sterling, 1992).

As I approach -- or surpass -- the age of my undergraduate professors, I find myself noting that the incoming students of today think that security problems began with them. They do not know of a pre-Internet era or of an age when information was not so readily and immediately

available. Many of the people new to the field of information security lack the historical or experiential perspective in the fields of information technology, computer science, and information assurance that might provide them a contextual framework with which to truly think outside the box and look at their defenses as their adversaries do.

My talk eventually became one of thoughtful reflection -- i.e., reminiscing about the "good old days." And in that retrospection of the 1970s and 1980s, I observed how the attack vectors on information today are different than in yesteryear, but many of the attacks are essentially the same. And, I believe, those analogies continue to exist today and that there might be some use in seeing how the attacks have matured and morphed over time.

The Information Age

It is hard to precisely nail down the beginning of the Information Age. There are those who say that it began in the 1980s with the proliferation of personal computers (PCs). Others suggest that it was the 1990s with the Internet and World Wide Web, while even others think that it was the last decade of social networking (Web 2.0) and the immediacy of communication that can topple governments that has ushered in the Information Age.

I grew up in southern California in the 1950s and came of age in the 1960s. As a child commuting to Disneyland, the House of the Future exhibit was a big draw, including neat appliances that no one yet imagined would be controlled by telecommunications networks. The AT&T exhibit of the Picturephone had lines of people as long as many of the rides. Dick Tracy comics every Sunday showed futuristic communications such as the 2-Way Wrist Radio, followed by the 2-Way Wrist TV. I saw my first computer in the mid-1960s -- a Honeywell mainframe at the University of Southern California. In my life, *that* was the heralding of the Information Age.

A more proper view might place the beginning of the Information age back when telecommunications began in the mid-1800s. But no matter when you want to believe that the Information Age started, one thing is consistently clear; to this day, we, at least in the U.S. -- as citizens, as consumers, as users of networks -- still do not take the protection of our information all that seriously. And what has become of our sense of privacy in an era when our youthful indiscretions are plastered on a global network that never forgets, exacerbated by the fact that we are the ones who often post the evidence of those indiscretions? And certainly the holders of our private information seem to believe that they own that information and we give the holders of that information almost free rein so that we can save a few dollars (Lane, 2009; O'Harrow, 2005).

Privacy and Social Security Numbers

My first experience with protecting my own private information came when I received my social security card as a teenager in the 1960s. I took seriously the admonition on the card that said "Not for identification" (Figure 1). Imagine my surprise when I went to college and found

that the social security number (SSN) was the primary identifying number, plastered all over everything from student identification cards to grade reports (Figure 2). The SSN was even on my employee badge at one of my summer jobs, working for a company that specialized in computing and information, although I took steps there to obscure the information (Figure 3).

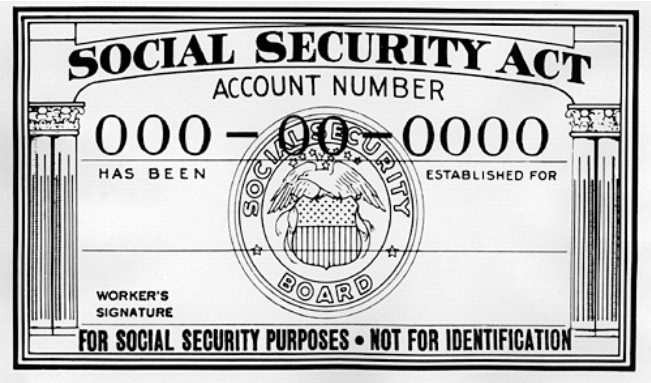

Figure 1. A sample social security card. Note the statement at the bottom, "Not for identification." *(Source: http://upload.wikimedia.org/wikipedia/commons/7/7c/Social_security_card.gif)*

Figure 2. Student identification cards from Humboldt State University (circa 1971) and the University of Arizona (circa 1975). Note the punch card motif in both, complete with cut left-bottom corner and cardpunch holes in the plastic that contained the SSN. Humboldt's card actually printed the the SSN twice.

Of course, things did not change very much at many colleges and universities over the decades except that the information leakage often became worse. My son, Joshua, for example, started his university career in 1999 at a school that assigned e-mail addresses in the format *jkessler9999@example.edu*, where the 9999 was actually the last four digits of his SSN. Joshua's SSN was assigned to him as a child while living in Vermont. Because of Vermont's small size, there are only two three-digit prefixes assigned to the state and the vast majority of people have the same first three numbers. Armed with Joshua's e-mail address and a tiny bit of personal knowledge would give a nefarious person a good guess at seven digits of his SSN and, I fear, it would be relatively easy for the remaining two digits to be socially engineered.

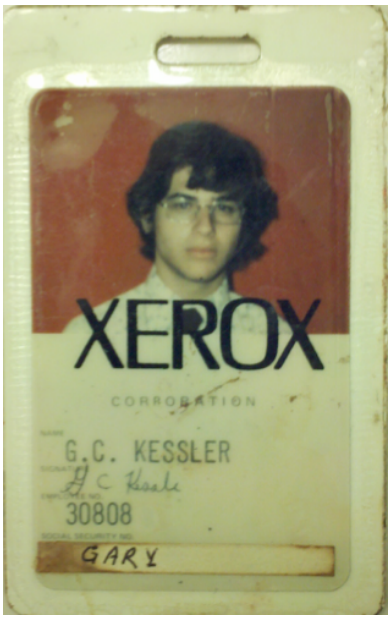

Figure 3. Employee identification badge from Xerox Data Systems (circa 1973). Note my first public effort at being a privacy advocate, where I put my name on a piece of tape over the SSN at the bottom of the badge.

When I contacted the Director of Information Technology at Joshua's college to complain about the use of the SSN in the e-mail address, he could not understand why I was concerned and, further, advised me that I was the only parent who had ever complained. He also told me that my son was required to have a university e-mail account and that this was the format. When I asked how international students who did not have an SSN got e-mail addresses, he responded that a dummy number was used. That option, however, was not available to U.S. students who had an SSN.

That school was not unique. At the college where I worked from 2000-2010, the SSN remained the primary identifying number until about 2005. Class rosters, admissions reports, and grade reports all had SSNs right there with student names and, in some cases, their date of birth. Many professors would post grades outside of their office using SSNs rather than students' names in order to maintain the privacy of the grade; given that the list was generated in alphabetical order rather than numerical order, however, it was pretty simple to figure out the name that matched the SSN (and the person's grade).

Computers and Paper Media

As an undergraduate mathematics major in the early-1970s -- before computer science degrees were common -- I discovered computers. Computers, or any of the automated calculating machines developed through the ages, were invented to solve problems while the solution still had relevance (Goldstine, 1972).

Consider the punch card tabulator. The counting for the 1880 U.S. census was completed in 1886 and it was estimated that the 1890 census would be finished in 1902, two years after the

1900 census would have begun. In response to this need, Herman Hollerith developed the punch card and automated tabulating machines; the 1890 census used this equipment and the counting was completed in 1892.

In World War II, one of the primary drivers for computers was for the preparation of ballistic tables. The problem to be solved was that manual computation methods could not always accommodate all of the necessary parameters in order to accurately ensure that ordnance hit its target before the target moved and a re-calculation would be necessary. Computers could perform what were considered to be complex calculations in seconds for which manual methods required many minutes, often rendering firing solutions obsolete. Thus the need for tables to aid in the field.

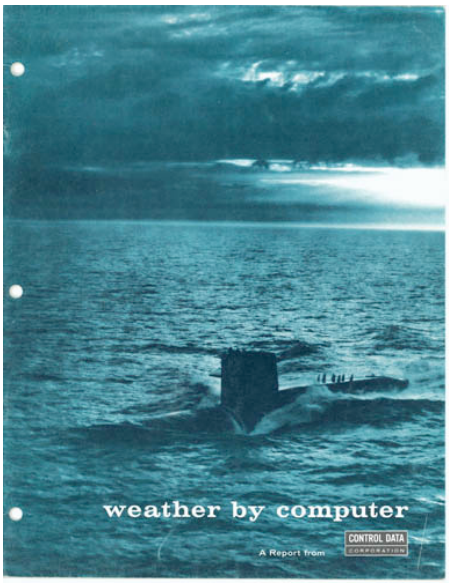

Figure 4. *Weather by Computer*, a report produced by Control Data Corporation. *(Source: http://archive.computerhistory.org/resources/text/CDC/CDC.WeatherbyComputer.1963.102641261.fc.lg.jpg)*

By the 1960s, computers were being looked to for other complex problems such as weather prediction, a holy grail of people for thousands of years (Figure 4). Here, again, the problem was to take all of the inputs and accurately predict the weather in a timely enough fashion to plan or react, as appropriate (CDC, 1963). And consider the computers used during the 1960s to aid in getting man to the moon.

Until the end of the 1960s, pretty much everyone who used a computer was a programmer. I took my first computer programming course in 1973 (Math 51, "Introduction to FORTRAN Programming") and eventually took every programming course that my college offered -- a half dozen or so, as I recall, not counting "Numerical Analysis."

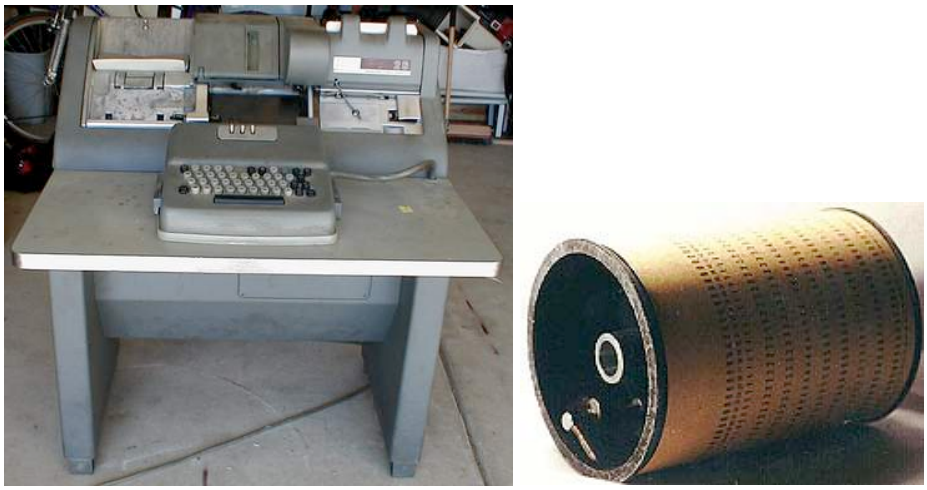

Figure 5. IBM 026 card punch machine (left) and the program drum (right). The program drum can be partially seen in the card punch on the left, behind the vertical slot above the keyboard. *(Sources -- Left: Sellam Ismail /Vintagetech.com [http://www.vintage.org/pictures /IBM%20026%20%28Front%29.JPG]; Right: Professor Gabriel Robins, University of Virginia [http://www.cs.virginia.edu/brochure/images/mus_135.jpg])*

The first computer that I used employed cards for input. A card deck was composed of some control cards (usually containing user information and indicating the language of the program that would follow), program code, and program data. Figure 5 shows an IBM 026 card punch.[1] To speed up typing, some users would code the program drum (Figure 5) to automatically skip to certain fields on the standard 80-column punch card (Figure 6).

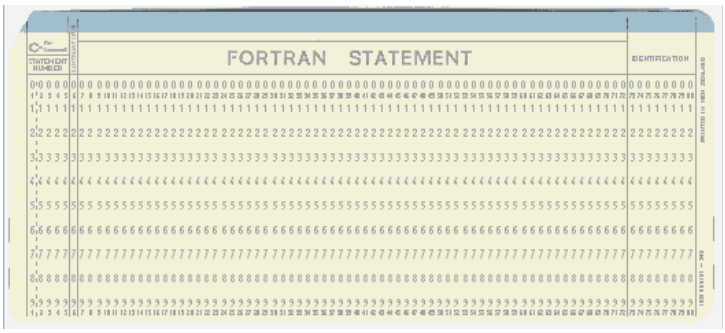

Figure 6. Typical 80-column punch card. *(Source: Douglas W. Jones, University of Iowa [http://www.cs.uiowa.edu/~jones/cards/collection/fortran.gif])*

It is hard to imagine the security issues related to punch cards but there certainly was hacking, ranging from the mischievous to the significant. Some of the more sophomoric incidents would be when someone would surreptitiously alter the program drum card so that it would put information in an incorrect field. As an example, the last eight characters on the card when writing a program in the FORmula TRANslation (FORTRAN) language were used to add sequence numbers to a card deck (in case the cards were dropped) and were ignored by the

[1] To this day, I can tell the difference between an 026 and 029 card punch from the side profile, which is a truly arcane skill.

compiler. By altering the program drum to start the sequence number in, say, column 72 instead of 73, a spurious character would be placed at the end of the program statement line, thus confusing the compiler and requiring that the programmer re-punch his/her card deck. Sometimes we would engage in social engineering, telling a new person in the programming lab that the printed labels on the cards actually had meaning to the computer's card reader, so, for example, that they had to use a card labeled FORTRAN STATEMENT when creating FORTRAN programs. These were silly antics, perhaps, but certainly in line with many of the ankle biters we see today in the hacker community.

The 1970s was alive with the hacking spirit back when there was no such thing as a Black Hat hacker. At that time, the only way to really learn system internals was to stretch a machine to its limits, often crashing the computer. Of course, when you crashed a mainframe, a lot of people noticed and their work disrupted; rebooting took many more minutes than restarting a Windows box today.

But more serious attacks on data occurred, as well. At my undergraduate college, everyone living in the dorms received a monthly telephone bill from Pacific Telephone. The bills came in the form of a punch card with your account number, phone number, and amount due. The card, which had only holes and no printing, needed to be returned with the payment. One year, a fellow student did a summer internship at PacTel and returned in the fall with a small stack of blank PACIFIC TELEPHONE punch cards. It was trivial to punch a new card with a new billing amount and it is still not clear to me how long this ruse worked for him.

Due to the large programs or data sets that programmers and researchers were creating at the time, users did not want to be walking around with huge boxes or trays of punch cards. For that reason, many of us kept our large card decks in the card punch room in the computer center. In some cases, very large data sets were transferred to magnetic tape (Figure 7). At that time, there was a sense of trust that others were not messing around or poking with your data although there was no real reason to believe that such trust was deserved or honored.

Figure 7. Old style databases: a box containing a set of punch cards (left) and magnetic tape (right). *(Sources -- Left: Arnold Reinhold [http://en.wikipedia.org /wiki/File:PunchCardDecks.agr.jpg]; Right: Professor Gabriel Robins, University of Virginia [http://www.cs.virginia.edu/brochure/images/mus_070.jpg])*

There were few, if any, protections to prevent an unauthorized person from accessing a box of cards in the computer center or even requesting that a particular tape be mounted so that a program could access it. Like today, someone can do a lot more damage by altering a database than they can by deleting data or destroying the entire database; these latter actions might be so obvious as to be detected by the owner who might then be able to take remediation steps to affect a repair. Minor alterations, however, might go undetected for a considerable amount of time, making a Bad Guy's small change ripple into big, untraceable, unfixable changes over time.

My Days With Batch Computers

The first computer that I used was in 1973, a Control Data Corporation (CDC) 3150 (Figure 8). The CDC 3150 was hand-wired and employed core memory. The capacity of core was limited to 131,072 24-bit words; i.e., it maxed out at less than 400KB of random access memory (RAM). This system predated virtual memory systems and paging, so the entire program, as well as data, shared the RAM space.

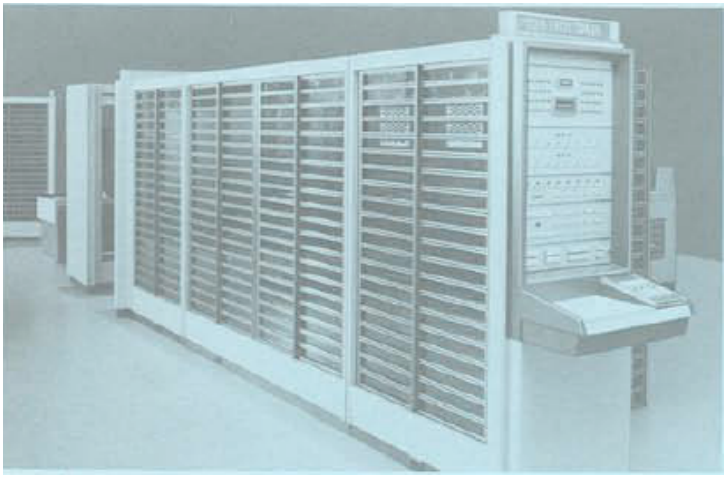

Figure 8. Control Data 3150 mainframe computer (circa 1972). *(Source: http://archive.computerhistory.org/resources/text/CDC/CDC.3000.1964.102646247.pdf)*

There were two interesting aspects of this system. First, the size of RAM limited the amount of data that the program could access. I crashed this system -- by accident -- when I created an array that required more space than the RAM could allocate because the system had no array bounds checking mechanism. While the impact of crashing the computer only directly hurt my program, it resulted in downtime for everybody as the system was rebooted. Since this "error" could be replicated at will, it was a denial-of-service (DoS) attack waiting to happen.

Second, since data and program code shared the RAM, a programmer could inadvertently -- or purposely, if that was the intent -- write programs with self-modifying code. Modern computers do not let you do this. Yet, we have similar attack vectors in the form of such events as SQL

Injects that, for all intents and purposes, allows an attacker to modify a query so as to gain unauthorized access to a database.

One of the next batch computers I used was as a graduate student in 1975, a CDC 7600 (Figure 9). The 7600 was a large, complex machine, so much so that CDC had technicians resident at the university's computer center. One of my fellow graduate computer science students found a vulnerability in the way in which the cards were processed.

Figure 9. CDC 7600 computer (internal wiring and dual-screen console).
(Source: http://en.wikipedia.org/wiki/File:Personable_Computer.jpg)

A typical card deck might look like the following:

```
!JOB
USER KESSLEGC
PASS SECRET
#FORTRAN
PROGRAM FOOBAZ
:
:
```

In this example, the first card indicates that this is the beginning of a new job (i.e., a new program). The next two cards provide the username and password of the user (note that one only had to peek at the `PASS` card to see someone else's password; this was much easier than the shoulder surfing the we had to do on the multi-user timesharing DECsystem10 that we also had at the school). The `#FORTRAN` card directs the computer to load the FORTRAN compiler; the next card is the first line of the actual program.

The 1970s was an era when programmers did whatever tricks were possible to save time; saving a millisecond here, a few nanoseconds there... whatever it took.[2] Someone observed that it required some hundreds of milliseconds to verify that the username and password were valid, yet thousands of milliseconds to load a compiler. Therefore, the reasoning went, why not set the card reader to read the first four cards in the card deck and process the `USER`, `PASS`, and `#` cards simultaneously, thus saving hundreds of milliseconds?

While this plan might appear to make sense, there was a flaw in the implementation. Specifically, the card reader would pause during the reading of cards while it was actually processing information. To implement the plan as described above, the system should have been set to read four cards and stop until two processes completed; namely, username/password validation and loading the compiler. Instead, the system read four cards but then kept reading if either process was finished. Again, the rationale was that the first line of a program could not be processed unless the compiler was loaded anyway, so no harm could come from that. That faulty reasoning was the vulnerability.

And the exploit of the vulnerability was pretty simple, as shown in the following card deck:

```
!JOB
USER STRAPPHA
PASS SECRET
#COMPASS
IDENT CRASH
:
:
```

In this example, the username and password are bogus which causes the system to spend the maximum amount of time searching the user database. The `#COMPASS` card loads not a language compiler but the 7600's assembler. Since the card processing was all handled by assembly language code, it took zero milliseconds to process this card. And, it was relatively simple to write a short (six to eight lines, as I recall) assembly language program that would crash the machine in less time than it took to detect that the username and password were invalid.

This flaw, by the way, was reported to the on-site CDC technicians. Their answer was, "Hmmm.... That's interesting. Don't do that."

This DoS exploit on the 7600 was a type of timing attack. These types of attacks still occur today. One common example is with gift cards available from many stores. In some stores, a gift card is activated as soon at the cashier enters the card into the system, even before the patron pays for it. Consider the following scenario. Alice and Bob are in a gift card theft ring. Alice goes

[2] Many students of the era may recall Grace Murray Hopper's lectures where she would carry with her handfuls of nanoseconds; i.e., one foot segments of wire, representing how far light travels in a billionth of a second.

into the Example Store and steals two gift cards. She then goes home where she copies the data from one card to the other, so that she now has two copies of the same card. She keeps one and gives one to Bob. Alice goes to a branch of the Example Store, stocks up on a lot of items, and goes to the cashier. She starts by giving the gift card to the cashier, asking that it be set at some value, say $250. She then gives the other items to the cashier. Near the end of this, she suddenly realizes that there's another item that she wants, asks the cashier to wait a moment, and sneaks out of the store.

Meanwhile, Bob is already ready at another branch of the Example Store. Upon some signal from Alice, he goes to a cashier, rings out, and pays with the gift card clone that has been activated at Alice's branch. This attack requires careful timing but certainly worked as recently as a few years ago.

My Early Days With Timesharing Systems

In the 1970s, timesharing was in its infancy. My undergraduate college was in northern California and we received timesharing services from another California state college in Los Angeles, nearly 800 miles away. Imagine playing Star Trek on a TTY Model 33 teletype machine at 110 baud; we seemed to have plenty of time and did not have the sensibility to care about all of the paper we were burning through.

During the 1970s, I had several opportunities to use Xerox (later Honeywell) Sigma computers, notably a Sigma 6 during a later bout in graduate school. The Sigma timesharing operating system was called CP/V and it afforded me with my first opportunity to purposely hack into a computer.

While using this system, I discovered what I thought was a vulnerability in the CP/V logout handler. When a user logged off their session, a two-line message such as the following was displayed on the terminal:

```
User kumquat logged out Thursday, Nov. 15, 1979 at 7:59 PM

Press RETURN to logon...
```

When a new user would approach the terminal and press the RETURN key, a message such as the following would appear:

```
UNIVERSITY OF VERMONT ACADEMIC COMPUTER CENTER
Thursday, Nov. 15, 1979 at 8:02 PM

Username:
```

After the username was supplied, the user would be prompted for their password. If there was a problem with the username/password, the system would provide an error message and again prompt the user to hit RETURN to get started.

Without much trouble, I wrote a logon spoofing program. I started a program that displayed a logout message such as the one above, and then walked away from the terminal. Another user almost immediately sat down and hit RETURN. My program displayed a welcome message and prompted for the username and password; the program then wrote those values to a file in my account, displayed a "Login error" message, and performed a terse logout (i.e., one where only the `Press RETURN...` portion of the logout message was displayed). The user then hit RETURN again and started over, this time with a valid system login message.

At this point, I sat down at another terminal, logged on, and, as expected, found the file with the user's name and password in my account. I immediately reported this to my friends in the Academic Computing Center and they told me, "Yeah, we figured that could happen. Don't do that."

Figure 10. Windows login dialogue box. Real or bogus?

Login spoofing attacks are still alive and well. One can walk into almost any college terminal and, upon hitting CTRL-ALT-DEL, find the familiar Windows logon screen popup (Figure 10). Very little would prevent a nefarious user from putting up a bogus login screen and capturing other users' login credentials.[3]

In 1981, I became the system manager of a Digital Equipment Corporation (DEC) VAX-11/750 running the VMS operating system (Figure 11). The VAX was one of the earliest mini-computer systems to employ a virtual memory architecture, where the system believed that it had 4GB of memory regardless of the actual physical capacity of RAM.[4]

One vulnerability with the VAX/VMS systems was that it was shipped with three built-in accounts:

- A system manager account named SYSTEM with a password of MANAGER

[3] It is true that CTRL-ALT-DEL is generally trapped by the operating system but there are other ways in which a bogus login dialogue box could be produced.

[4] I should note that it was common, at the time, for a mini-computer system to have several MB of physical RAM, so a 4 GB (32-bit) address space was considered huge.

- A field service account named FIELD with a password of SERVICE
- A default user account named USER with a null password

Figure 11. Digital Equipment Corporation VAX/11-750. *(Source: http://www.museumwaalsdorp.nl/computer/images/11750cpu.jpg)*

All system managers are told in system management training to:

- Change the password on the system manager account and only logon as SYSTEM when performing tasks that required that level of privilege.
- Change the password on the field service account and disable the account except when field service personnel are on the premises.
- Add a password to the default user account and disable it, using it only as a template for adding new user accounts.

The Cuckoo's Egg (Stoll, 1989) nicely describes one scenario when system managers do not adhere to these simple rules and the particular vulnerabilities when these systems are networked together. Even today, system managers use simple passwords, default passwords, and/or the same password on multiple systems. Users are not trained in good password strategies and management, and most system managers learn information security as an add-on to their information technology (IT) function rather than integrated into it. A large number of passwords today remain guessable, if not neatly written on a piece of paper taped to the monitor, bulletin board, or underneath the keyboard.[5]

[5] Another case in point: In the summer of 2010, I performed a site review for a recreational park. The ticket booth had several computers and all users shared the same password. Because the password was changed frequently, ostensibly for security purposes, it was written on the piece of paper hanging above one of the ticket windows. Because of the orientation of the building, any causal observer, including customers, could read the username format and the password.

Remote Access

Modems were developed in the early-1960s although they did not start to see widespread use until the development of portable terminals and, of course, personal computers (PCs). My first exposure to modems was in 1977, when I was "allowed" to borrow a Texas Instruments (TI) Silent 700 terminal to access the work mainframe from home (Figure 12). My first foray into telecommuting made it clear to that this was a way to have people work 24x7 and was a harbinger of work-at-home technologies to follow, such as the Integrated Services Digital Network (ISDN), cable modems, and digital subscriber line (DSL) technologies.

Modems present vulnerabilities because many sites do not carefully deploy them. War dialers (as highlighted in one of my favorite movies, *War Games*) are still used today although there are fewer modems to find. The lines themselves, and even the systems connected to them, are not always as secure as they might be. The infamous "attack" on Bell South's 911 system in 1988 was not nearly as sophisticated as one might have thought. A small group of users on the 911 team had a computer system on which they did their work. Because they were a small team with a high level of trust, they never secured the accounts with passwords. At some point, they decided to put a modem on the system so that they could have remote access to the computer. Since the number was not public, the reasoning apparently went, no one will know to dial up and so passwords were not assigned to the accounts. A hacker group using a war dialer stumbled upon the modem, dialed in, and accessed one of the accounts. Naturally they downloaded every file they could find.

Figure 12. Texas Instruments Silent 700 terminal with acoustic coupler and thermal printer. *(Source: http://www.flickr.com/photos/hckhckhck/124699193)*

Hackers and information security analysts today employ a slightly different version of war dialer when probing networks; we now use port scanners to find listening ports on systems connected to the Internet and wireless network scanners to find unsecured wireless local area networks (WLANs). The methods may be a bit more sophisticated today but the concept is the same; scan the landscape for anything that will communicate back with you, learn what you can about it, probe it for vulnerabilities, and then exploit the system if you can.

Enter PCs

The first small-scale computer that I put together and wrote code for was in 1981; a DEC LSI-11/23 (Figure 13), which was a "kit" version of DEC's PDP-11/23. This was a very powerful single-user system, running a real-time operating system called RT-11[6] and supporting programming languages such as FORTRAN and Pascal.

Figure 13. Inside view of an LSI-11/23. *(Source: http://en.wikipedia.org/wiki/File:DEC_LSI11-23.jpg)*

Figure 14. Apple][+ system with dual floppy disks. *(Source: http://www.uelectronics.info/sites/uelectronics.info/files/images/apple_ii_large.jpg)*

[6] A popular DEC t-shirt at the time --which, of course, I owned -- read, "For a good real time -- RT-11."

PCs for the home started to become available in the mid-1970s, mostly as kits for hobbyists. Most of these systems required that the user/builder write code in some form of assembler or machine language. With the introduction of a Beginner's All-purpose Symbolic Instruction Code (BASIC) interpreter, more people could write -- or buy -- programs for their PCs and the market grew. The first PC-class system that I owned was an Apple][+ (Figure 14), purchased in 1981, around the time that the IBM PC was introduced.

These systems were pretty much stand-alone systems. Some information could be exchanged between users via modem, if the two users had them and the local telephone lines were quiet enough to support even the 300 baud connections that were then available. Software could also be exchanged via third-party Bulletin Board Systems (BBS), rudimentary precursors to Internet-style file sharing sites. And like today, there were risks at these sites. Viruses were not yet rampant in the early 1980s, but one never knew what hidden code might be in a program that was acquired from a BBS or any other source.

In 1981, I published some BASIC date manipulation routines in a wonderful hobbyist magazine called *Creative Computing* (Kessler, 1981). These routines, exactly as published, somehow found their way into a commercial product without any attribution. Since I had published the code in a magazine, I had obviously placed the routines in the public domain. What mattered here was only that the author of the commercial software had testified under oath (due to a theft of intellectual property lawsuit brought by someone else) that he had written every line of code himself. This all happened before the open source and free software movements were in any way formalized. I never found out how it was discovered that my routines appeared in this particular program; it was obvious that the date routines were written by someone other than the author of the rest of the program but how it was linked back to me remains a mystery.

Nevertheless, I often think about these issues when I use any open source materials myself. The great part about open source software, of course, is that any user can read the code, ostensibly to validate its functionality. But, was this not the same argument behind the development of the COmmon Business-Oriented Language (COBOL) in the late-1950s? Managers, it was thought, could -- and would -- be able to read programs because instead of programmers using obscure instructions such as:

```
PMT = AMT * INT
```

they would write:

```
MULTIPLY AMOUNT BY INTEREST GIVING PAYMENT
```

COBOL's syntax may have made each *line* of a program easier to understand but it did not really make *programs* easier to understand. And how many users today actually read the open source code and truly understand what each line is doing prior to installing the software?

Mainframe computers provided a means whereby many users could share a single computing resource. From a security and management perspective, it meant a single system manager, a single set of security policies, and a single security manager. For many good technical reasons, we saw a rise in the use of minicomputers in the 1970s and microcomputers in the 1980s; we saw lower costs and a higher amount of dedicated compute power delivered to end user. But it now meant that every user became, in effect, a system and security manager, a task for which they were not trained.

The 1980s also saw a rise in new LAN standard technologies such as Ethernet, IEEE 802.3, and IEEE 802.5 Token Rings. As distributed computing became the industry trend, it became harder to engage in centralized management of systems. Indeed, security became a major issue in this era as there were so many different ways to break into a corporate network. While infosec professionals were not surprised, it was the late-1980s when computer security started to enter the popular media; both the Internet Worm (1988) and Michelangelo virus (1991) made evening national news broadcasts (Randall, 1997). It was not until the 1990s that centralized domain servers started to appear that would allow a network manager to remotely push updates and security policies to networked computers, bringing us back full circle to centralized management -- at least for a while.

Figure 15. TRS-80 Model 100 laptop computer. *(Source: http://oldcomputers.net/trs100.html)*

My first portable/laptop computer was a TRS-80 Model 100 in 1984 (Figure 15). While not the most powerful system in the world, it became the first computer that I could accidently leave in an airport lounge in Chicago. Having no inherent security, anyone could turn this computer on and read the contents. It was also the first computer I owned that had a built-in modem.

The Age of Networking

One of my earliest recollections of a newspaper picture -- I was just four years old -- was the Russian launch of Sputnik in 1957. One of the direct responses to Sputnik by the U.S. was the creation of the Department of Defense's Advanced Research Project Agency (APRA). ARPA's

greatest creation from a network perspective was the ARPANET. Starting with four nodes in 1969 (Figure 16), this network, of course, morphed into today's Internet.

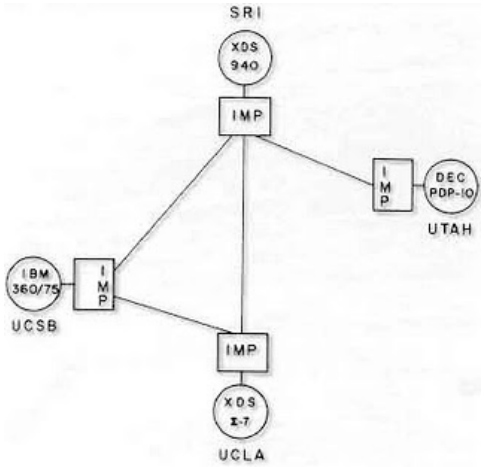

Figure 16. The ARPANET's first four nodes in 1969. *(Source: http://www.facstaff.bucknell.edu/mvigeant/univ_270_07 /internet/beginning.asp)*

I first worked on the ARPANET in 1981 when I worked at one of the national laboratories with a connection to the network that, by this time, had more than 200 nodes. At that time, we were happy to share a text file or logon to a remote terminal. A 56-kbps link was considered to be lightning fast and it was, when compared to our needs.

Neither the ARPANET nor the Transmission Control Protocol/Internet Protocol (TCP/IP) suite were built with security in mind. One reason was the fact that the network was closed and the community small. The development of the Internet in 1986, gateways to external network users in the late-1980s, and open commercialism in the early-1990s changed everything. While the Internet Worm may have belied the trust that we had in the members of the Internet community, the Green Card Lawyers showed, in 1994, that many new users of the Net neither knew of nor cared about politeness on the Internet. It was the beginning of an age when people would act in cyberspace any way they pleased merely because they could (Hafner & Lyon, 1996).

Spam has been rampant on the Internet almost since it's inception but is hardly a new concept; junk mail has been around for decades. Nevertheless, even the spam that we get is just a variance of older scams. Consider Nigerian 419 spam messages. Prior to the Internet and e-mail, facsimile machines were the vectors of 419 messages. And who had faxes? Mostly doctors, lawyers, and other professionals, and these were the perfect targets for the purveyors of these messages because these recipients were perceived to have money. Before faxes, 419 spammers sent letters, again targeting their posts to well-paid professionals or those living in up-scale neighborhoods. Although decidedly low-tech, the senders of these messages were using some of the same skills as the spear-phishers of today.

By the mid-1990s, residential Internet service became a reality with the rollout of cable modems and DSL. Residential hub/routers became available so that people were able to build affordable residential LANs to interconnect to the Internet. In the early days of residential Internet service, many cable and DSL providers only allowed a single device to connect to the service. To enforce this, the connected system's Medium Access Control (MAC) address was configured at the network ingress point. Residential routers since the beginning have allowed MAC address spoofing so that the router could be assigned the registered MAC address.

But with the introduction of residential networks, everybody now became a network, security, and firewall manager. The introduction of wireless LANs in the late-1990s just exacerbated the fact that most people at home had no idea how to secure their network and most did not care, reasoning that they did not have any information that a Bad Guy would want.

This attitude was often the same in the corporate environment, as well. I configured my first firewall in 1994 and management at the company at which I worked for observed that a) we were in Vermont where I did not lock my car doors in the parking lot, so why lock the door to the server room and configure a firewall for the router, and b) what would anyone want to steal from our company, anyway?

As years went by, the industry realized that a single network firewall, alone, was not sufficient. To adequately protect our assets, we need a network-based firewall, server-based host firewalls, intrusion detection systems, intrusion prevention systems, vulnerability scanners, and more -- what we call *defense-in-depth*. I used to add in one more feature, namely, that the vendor of these various products should be as different as possible, introducing *biodiversity*. Both of these characteristics, of course, have models in the real world. Castles, for example, were generally more than communities with high walls; they also had moats, mountains, flat ground, guard patrols, and other characteristics that made it hard for an enemy to easily break in. And biodiversity has saved more than one forest or crop from total devastation from a single bug. Obviously, these concept are not new.

And attacks on computers and networks take similar advantage of the features of cyber defense systems as physical attacks do on physical defenses. In a large area that needs protection, for example, an intruder might gain valuable intelligence merely by throwing some stones at a remote sensor and noting how long a response takes. Constantly stimulating the same sensor might eventually result in a delayed response, as the protectors start to believe that there is a problem with the sensor. Similar thinking takes place with the automated responses of cyber systems. As an attacker, if I can learn how your system will respond to particular stimuli, I can tailor my attacks to provoke the response that I want.

Networks and Privacy

Social networking had its roots at a major university in Cambridge, Massachusetts in the early part of the decade. But that university was the Massachusetts Institute of Technology (MIT), not Harvard, and it was the early-1960s, not the early-2000s. Dr. J.C.R. Licklider (Figure 17), an

MIT professor, had degrees in mathematics, physics, and psychology. In 1962, while at Bolt, Beranek and Newman (BBN), Licklider started to describe his concept of the Intergalactic Computer Network, a global communications network that would allow non-technical people around the world to communicate with each other, unfettered by governments and national boundaries. Of course, at the time he had this idea, there was no infrastructure on which to build this dream; his ideas pre-dated the existence of packet switching, the underlying technology on which the ARPANET was built seven years later. BBN, of course, was the company that built that network. Dr. Licklider did not merely take the ideas of others and make them better; he had the vision before anyone else (Hafner & Lyon, 1996; Salus, 1995).

Figure 17. Joseph Carl Robnett Licklider (1915-1990). *(Source: http://scopeweb.mit.edu/wp-content/uploads/2008/03/jcl2.JPG)*

In the 1970s and 1980s, the users of the ARPANET were techies whose idea of a social network was an e-mail with artwork based on American Standard Code for Information Interchange (ASCII) printable characters (Figure 18), an emoticon, or a cool signature block. Before Facebook and MySpace, we had the Name/Finger protocol, and the accompanying PLAN and PROJECT files. Then we had our Web pages.

The ARPANET was a closed network, however, and you truly did know almost everyone else on the 'Net. Today's social networks provide the opportunity for an incredible assault on one's privacy, targeting a demographic that interprets the imperatives of privacy very differently than my own generation. Sharing information with 3.5 billion of your nearest and dearest friends does not seem to deter many people because they are getting some service for "free" -- as far as they can tell. And so many people voluntarily let other people know their business, such as services that will display one's movements in real-time and those who advertise where and when they are traveling. The Web site *pleaserobme.com* has called attention to this over-sharing of information but it is unclear that people are deterred.

Twenty-plus years ago, I controlled exactly how much information about myself was posted on the Internet. Today, the incredibly powerful linkages between Web sites, social networking

sites, Internet archival sites, and more means that I have next to no control over the dissemination and lifetime of "my" information.

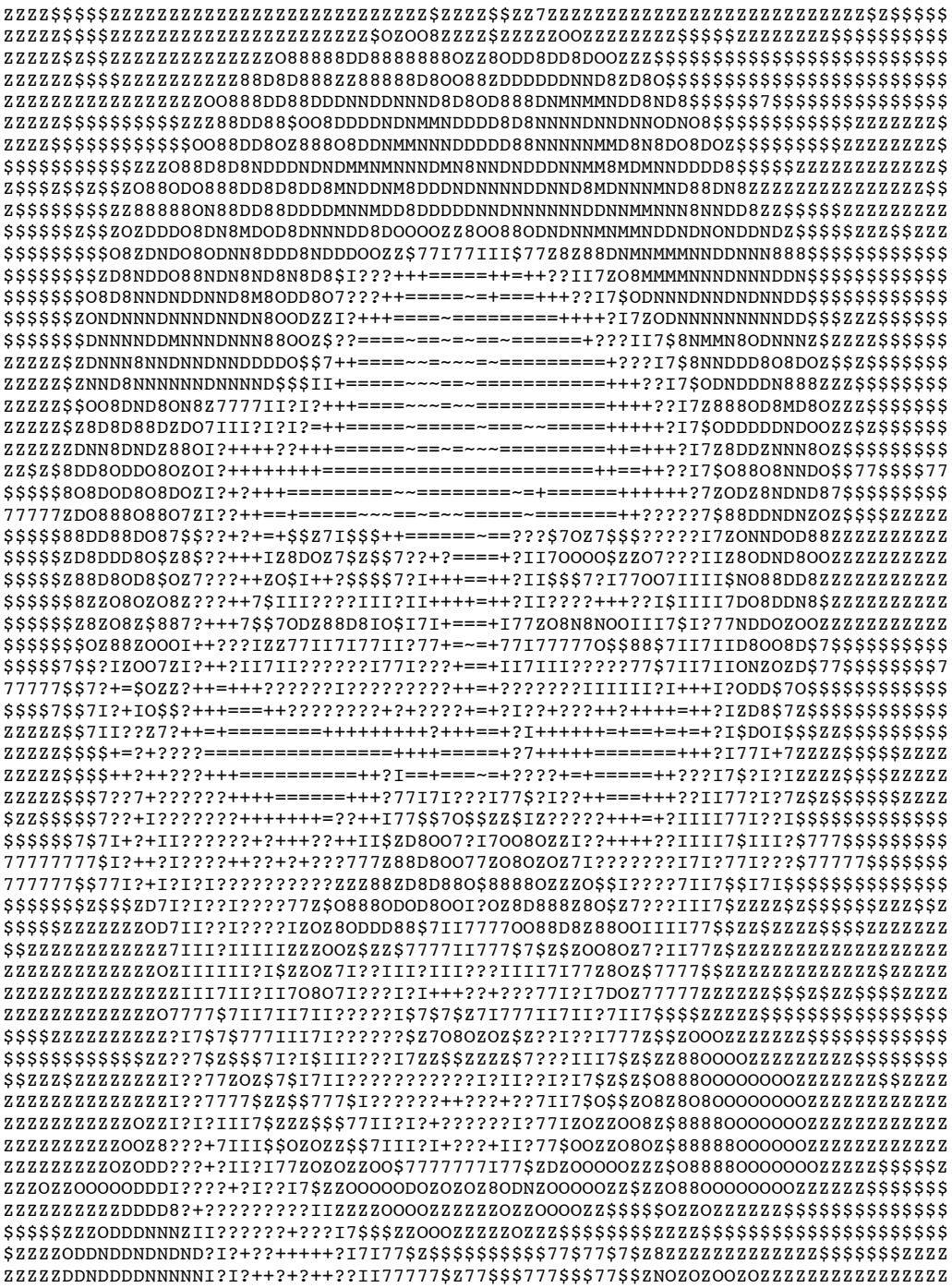

Figure 18. A portrait of the author in ASCII art form (from *GlassGiant.com*).

Sun Microsystem Chief Executive Officer (CEO) Scott McNealy said, in 1999, "You have zero privacy anyway. Get over it" (Sprenger, 1999). This attitude has morphed today, as reflected by Facebook CEO Mark Zuckerberg, into trying "to make the world more open and connected"

(Zuckerberg, 2009). This sounds noble, perhaps, but it has just made every user into -- yes, you guessed it -- an information security manager.

Final Observations

Mainframe computers, networked mainframes, personal computers, LANs, the Internet, the Web, WLANS, social networks... Each has provided unique vulnerabilities, exploits, and attack vectors, but the types of problems remain interestingly similar. This is not to say that there are no new problems and threats, but is meant to suggest that not everything is new.

It could be argued that any information security threat fits into a taxonomy built around the confidentiality, integrity, and availability of information, also known as the *CIA triad*. Some choose to expand this to the so-called *Parkerian hexad*, which addresses the confidentiality, possession, integrity, authenticity, availability, and utility of information (Parker, 2009). Regardless of your model, the problems can clearly be whittled down to a finite number of classifications.

It is imperative that information assurance professionals understand the threat vectors and even be able to conjure up vectors of their own in order to successfully build defenses. Thinking like your adversary is as fundamental to the defense of your assets as it is to warfare, which is why so many security sources quote Sun Tzu's *The Art of War*; interestingly, many fundamentals of modern warfare have an analog to a document that is more than 2,500 years old (Watson, 2011). It is not so startling, then, to think that today's threats on information might have analogs to the beginning of an Information Age that is only 150 years old.

Focus, then, seems to be the order of the day for protecting our information. Many sources have observed that airport security in most countries is weak because it focuses on finding bombs rather than bombers (Baum, 2010). In that spirit, I would suggest -- as others have -- that we concentrate on protecting the information and the information flow rather than focusing on the computers and the networks.

About The Author

Gary C. Kessler, Ph.D., CCE, CISSP is the president of Gary Kessler Associates, a training and consulting firm in the area of computer and network forensics and security. He is also editor-in-chief of the *Journal of Digital Forensics, Security and Law*; a member of the Vermont Internet Crimes Against Children (ICAC) Task Force; an Adjunct Associate Professor at Edith Cowan University in Perth, Western Australia; and Associate Professor and program director in the M.S. in Information Assurance program at Norwich University in Northfield, Vermont.

References

Baum, C. (2010, November 29). Follow the Israelis; Look for the Bomber, Not the Bomb. *The Salt Lake Tribune*. Retrieved March 8, 2011, from http://www.sltrib.com/sltrib/opinion/50770208-82/security-bomber-airport-israel.html.csp

Control Data Corporation (CDC). (1963). *Weather by Computer*. Minneapolis: Control Data Corporation. Retrieved March 1, 2011, from http://archive.computerhistory.org/resources/text/CDC/CDC.WeatherbyComputer.1963.102641261.pdf

Goldstine, H.H. (1972). *The Computer: From Pascal to von Neumann*. Princeton, NJ: Princeton University Press.

Hafner, K., & Lyon, M. (1996). *Where Wizards Stay Up Late: The Origins of the Internet*. New York: Simon and Schuster.

Kessler, G.C. (1981, December). Want a Date? A Collection of Date Manipulation Algorithms. *Creative Computing*, *7*(12), 214-218.

Lane, F.S. (2009). *American Privacy: The 400-Year History of Our Most Contested Right*. Boston: Beacon Press.

O'Harrow, Jr., R. (2005). *No Place to Hide: Behind the Scenes of Our Emerging Surveillance Society*. New York: Free Press.

Parker, D.B. (2009). Toward a New Framework for Information Security. In S. Bosworth, M.E. Kabay, & E. Whyne (eds.), *Computer Security Handbook*, 5th ed., Volume 1, Chapter 3. Hoboken, NJ: John Wiley & Sons.

Randall, N. (1997). *The Soul of the Internet: Net Gods, Netizens, and The Wiring of The World*. London: International Thompson Computer Press.

Salus, P.H. (1995). *Casting the Net: From ARPANET to INTERNET and Beyond*. Reading, MA: Addison-Wesley.

Sprenger, P. (1999, January 26). Sun on Privacy: 'Get Over it.' *WIRED*. Retrieved March 8, 2011, from http://www.wired.com/politics/law/news/1999/01/17538

Sterling, B. (1992). *The Hacker Crackdown: Law and Disorder on the Electronic Frontier*. New York: Bantam Books.

Stoll, C. (1989). *The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage*. New York: Pocket Books.

Watson, J. (2011). Sun Tzu's Art of War. Retrieved March 8, 2011, from http://suntzusaid.com/

Zuckerberg, M. (2009, December 1). An Open Letter From Facebook Founder Mark Zuckerberg. *The Facebook Blog*. Retrieved March 8, 2011, from http://blog.facebook.com/blog.php ?post=190423927130