Reaction: The Internet Security Paradox

John McHugh
Canada Research Chair in Privacy and Security
Director, Privacy and Security Laboratory
Dalhousie University
mchugh@cs.dal.ca

DALHOUSIE UNIVERSITY Inspiring Minds
Faculty of Computer Science
Privacy and Security Lab

A poke in the eye ...

My first experience with computer security came in the late 1960s when the resident customer engineer for the IBM 360/40 I worked on gave me a list of about a dozen ways to execute a user program in supervisor mode with a protection key of zero.

The first one that I tried crashed the computation center computer at George Washington University in an interesting way.

Working for a number of years as a scientific applications programmer, I discovered a number of ways to make machines do what I needed to do to get my job done, but never thought seriously about security until ...

... (with a sharp stick)

…I became interested in program verification in the late 1970s. Joining the Gypsy group at Texas, I fell under the (financial) spell of the NSA, the only group funding verification research at that time.

Using verification to develop systems that would be difficult to compromise seemed to be a noble cause, but it didn't work out too well, and we have gone on to other things.

In November of 1988, the Internet (such as it was) got a rude awakening. Someone had released a self replicating code that crashed some machines and drastically slowed others. OUCH!!!

DALHOUSIE UNIVERSITY Inspiring Minds
Faculty of Computer Science
Privacy and Security Lab

Surprised? Who? ME??

Evidence seemed to point to a graduate student at Cornell, Robert T. Morris, as the likely culprit. He was eventually convicted and fined. When I recalled a conversation that George Dinolt, then of the then Ford Aerospace Corp. and I had with Morris's father in August of 1988, I was not surprised.

Being generally optimistic, I am surprised by the fact that the same sorts situations that allowed the Morris worm to propagate in 1988 are still prevalent today. In my naïveté, I expected us to do better.

At the risk of sounding whiney, I'm going to talk about this.

Reactions to Morris

At the time of Morris, the internet was still primarily a research and academic endeavor. NSA and DARPA were the primary drivers of computer security research. The DARPA program manager for much of this work was Bill Scherlis, a faculty member at CMU. DARPA decided to create an organization to deal with future incidents of this type. DARPA operated the Software Engineering Institute, conveniently located at CMU, and CERT was born. In the early days, the CERT could provide direct assistance to almost anyone who experienced an attack on their computer.

DALHOUSIE UNIVERSITY Inspiring Minds
Faculty of Computer Science
Privacy and Security Lab

The very name is reactive

- CERT initially stood for Computer Emergency **Response** Team. It is now a registered service mark, and there are numerous similarly styled organizations throughout the world.
- While reaction is definitely called for in the wake of a security incident, it is not clear that reactive approaches address the root causes of our problems.
- If we reexamine the Morris worm, we see that it had very little to do with security, per se.

Why the "Morris Worm" worked

- The successful propagation of the worm was the result of a mix of
  - Misplaced trust (1 general case). Poor security practice aggravated this case.
  - Poor software engineering (2 cases)

DALHOUSIE UNIVERSITY Inspiring Minds
Faculty of Computer Science
Privacy and Security Lab

Misplaced trust.

- Unix systems have a notion of mutual trust.
  - globally in /etc/hosts.equiv
  - per user in .rhosts
- It is possible to configure systems so that a user logged on to one machine need not give a password to access a trusted peer.
- By cracking weak passwords, the Morris worm was able to reach (and often infect) many machines using this mechanism.
- One can argue that these are security issues, but they also relate to complexity of operation.

To share is golden, to err is human

- Networking is all about sharing. In the early days, we were all friends and knew our friends would not harm us. Much of the growth of the internet has been driven by a desire to share, whether it be for a fee or free.
- Unfortunately, in many cases the pursuit of convenience in sharing has been at the expense of any way to set and enforce limits.
- One might argue that not enough people have been hurt to create a consensus for effective controls on sharing. Most individuals and businesses see the benefits as outweighing the risks.

DALHOUSIE UNIVERSITY Inspiring Minds
Faculty of Computer Science
Privacy and Security Lab

Software Engineering Failures

- The worm became "aware" of other machines because they were mentioned in various files on the infected host. This allowed two other attacks
  - A buffer overflow exploit against *fingerd*

    A message was constructed that was too big for the array the program used to hold it. This caused code in the message to be executed.
  - A misconfiguration exploit against *sendmail*

    Commands sent to the attacked host were executed there.
- Both allowed the worm a foothold on another host.

Buffer overflows are avoidable!!!

- The general solution of the buffer overflow problem requires the programmer (or programming environment) to reason about data sizes. This requires a combination of type information and input checking but has a low run time cost.
- The problem can also be solved by checking data structure references for legality at run time or by using “type safe” languages such as Java rather than unsafe languages such as C
- In general, *Defensive Programming* covers this area.

DALHOUSIE UNIVERSITY Inspiring Minds
Faculty of Computer Science
Privacy and Security Lab

One root of the problem - bad education

- Many software engineering texts teach a contract model of programming.
  - The developer of a routine publishes an interface specification.
  - If the specification is honored by the user, a specific result is guaranteed.
  - If the specification is not honored, the results are undefined.
- In the hands of honorable and conscientious participants, all is well, but deliberate contract violations lead to things like buffer overflows. So does carelessness.

The wages of complexity are chaos!

- The misconfiguration problem is more subtle. The sendmail program was so complex that trial and error was the rule. As Spafford noted

  "Stories are often related about how system administrators will attempt to write new device drivers or otherwise modify the kernel of the OS, yet they will not willingly attempt to modify sendmail or its configuration files."
- The failure to design programs so that they can be used easily, safely, and securely is a failure in the "human factors" part of software engineering.

DALHOUSIE UNIVERSITY Inspiring Minds
Faculty of Computer Science
Privacy and Security Lab

Fast forward ... Nearly 18 years later

- One would think these problems would have been fixed, but consider:
  - Vul Note VU#394444MS Hyperlink Object Library stack buffer overflow
  - Vul Note VU#159220MS IE heap overflow
  - Vul Note VU#988356 Mac OS X stack-based buffer overflow via specially crafted TIFF file
  - and others in the past few months.
  - Note also that misplaced trust via file shares and URLs is also a problem

Unmanageable complexity

- As efforts are made to make operating systems more robust, attackers are turning to applications
- Applications typically have access to anything the user has and this is enough to do damage!
- The tendency to do everything on the web has greatly increased the complexity of web browsers and the applications they implicitly invoke.
- http://browserfun.blogspot.com/ is publishing a browser vulnerability per day during July.
- A quick glance at the list of BlackHat briefings shows a large number of sessions on applications

DALHOUSIE UNIVERSITY Inspiring Minds
Faculty of Computer Science
Privacy and Security Lab

Why can't we get ahead?

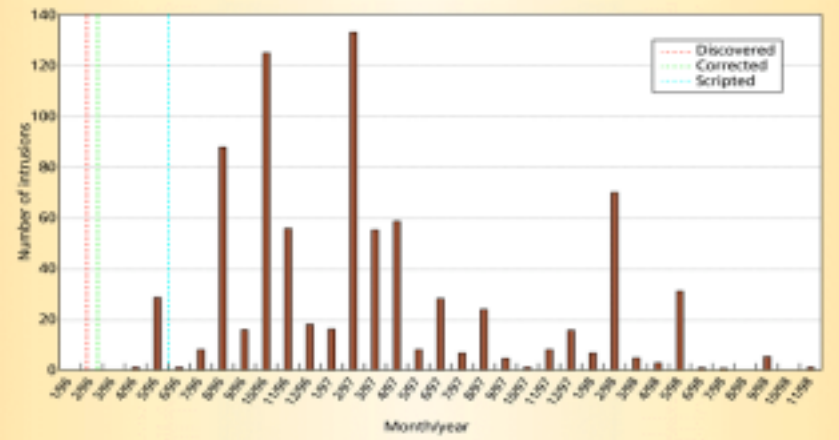
- The adversaries have two things going for them
  - Automation - Scripting in various forms reduces the skill level needed top perform an exploit.
    - With misplaced trust, the victim oftens performs the exploit
  - Laziness - We don't fix the fixable
  - Before returning to the general theme, lets look at a few interesting cases

Evolution (attacks), Devolution (attackers)

This old, but still the trend continues

- Scripting and automation continue to reduce the skill levels associated with sophisticated exploits.
- Users are often persuaded to perform exploits with the promise of something of value. Spyware is often the residue.
- As applications increasingly do “useful” things without being asked, the opportunities for scripted exploits increase.
- Systems are too complex for users to give meaningful guidance

phf incident history – from Arbaugh'00

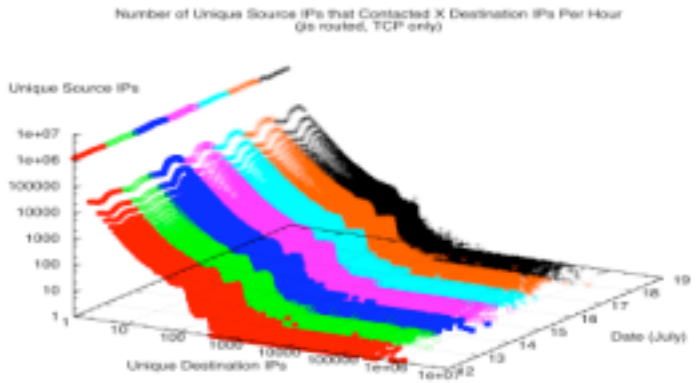


DALHOUSIE UNIVERSITY Inspiring Minds
Faculty of Computer Science
Privacy and Security Lab

This is also dated, but ...

- There is a school of thought that believes patch availability is the key to stopping the spread of exploits.
- Using CERT data from a few years ago, we found that the patch was often available before (long before) exploits started to appear.
- The script (or other automation) was key. The phf exploit apparently became passé, then resurfaced.
- We have seen an acceleration in vul - exploit timings, so this is probably not valid for many recent cases, but it provides food for thought.
- Worms are persistent.

Outside to inside - July 2003



DALHOUSIE UNIVERSITY Inspiring Minds
Faculty of Computer Science
Privacy and Security Lab

Internet wide disturbance

- The ripple in what would otherwise be a fairly straight log/log plot of connectivity was observed from at least Jan - Aug 2003.
- It went away when Blaster appeared in Aug 2003.
- A similar ripple existed from Feb 11 to May 31 2004 coinciding with the lifetime of Welchia-B
  - In this case, the ripple is due to a few hundred machines scanning at a low, fixed, rate induced by a loop with a “sleep” system call to induce a fixed scanning rate.
- In both cases, they persisted until killed, not patched.

Back to reaction.

- Every once in a while, the internet gets another poke in the eye. Some of the more significant ones:
  - Melissa in (March1999)
  - Code red (July 2001)
  - Slammer (January 2003)
  - Blaster (August 2003)
  - Etc.
  - DDoS attacks (seen 1999, widespread 2000+)
- A look at the advisories shows reactive responses.

DALHOUSIE UNIVERSITY Inspiring Minds
Faculty of Computer Science
Privacy and Security Lab

Reactive responses - 1

- For the worms and viruses, many of the “fixes” are similar.
  - Disable macros, block ports, etc.
  - These are inherently reactive, and they reflect the lack of a meaningful “security posture” for the pre-infection victim.
  - Note that a conservative - install and enable only what you really need deployment approach - would greatly reduce the ability of these malcodes to propagate.

Reactive responses - 2

- DDoS attacks provoked a different reaction. A variety of approaches are used.
  - Resource hardening - provision network and servers to handle loads - Akamai, etc.
  - Selective blocking / traffic modification
  - Traceback mechanisms to identify sources
- Attackers are using `bot networks with thousands of attackers and an “arms race” is in progress.
- Many attacks are combined with extortion attempts and victims are often sites with marginal ability to enlist law enforcement, e.g. pornography and gambling.

DALHOUSIE UNIVERSITY Inspiring Minds
Faculty of Computer Science
Privacy and Security Lab

And so it continues

- The general notion of DDoS is to create an unmanageable workload for the victim while keeping the attacker workload manageable.
  - Distributing the attack workload is one way - hence the value of large `bot nets to attackers
  - Amplifying effects is another - SMURF amplified senders; recursive DNS attacks amplify volume.
  - `Bots triggering amplifiers can produce Gb rate attacks.
  - But amplification is yet another example of misplaced trust (and possibly a reaction to complexity)

So, what can we do?

- When I started doing research in security, the goal of our sponsors was to produce systems that were very difficult to compromise - "provably secure"
- By the mid 1990s, we had learned enough to start to have modest success, but in limited ways
- In the mean time, the PC arrived and matured from a toy to a tool (without giving up its childish ways - Financial Times speculation on post Gates Microsoft sometime around 11 or 12 July 2006)
- The customers voted with their pocketbooks, and the goal of perfection (perceived to be at the expense of utility) was largely abandoned.
- **We gave up a quest for perfection, but did we have to get such a bad alternative?**

DALHOUSIE UNIVERSITY Inspiring Minds
Faculty of Computer Science
Privacy and Security Lab

Could we be proactive? - 1

- Staying ahead of attacks.
  - Despite the fact that we have seen some instances where the vul - exploit interval is very small, there is a residue of published vulnerabilities awaiting exploits.
  - We should be working to try to correct the underlying problems with routing, DNS, etc.
  - The community also needs to put on their black hats and develop analytical frameworks that can predict attack trends and potential consequences.

Could we be proactive? - 2

- Can we change development attitudes?
  - Somehow, the creation of internet applications needs to be seen as a craft with pride of ownership.
    - "779 days without a buffer overflow" to paraphrase the signs at industrial plant entrances.
  - Teach defensive programming. Repudiate the contract model. MS is trying, to give them credit.
  - Revoke signing keys for compromised applications.
    - Ohh, that's right, trusted has nothing to do with trustworthy
    - And I'm not sure that revocation is well supported.

DALHOUSIE UNIVERSITY Inspiring Minds
Faculty of Computer Science
Privacy and Security Lab

Could we be proactive? - 3

- Can we be a little less nerdy!
  - Surely, we could put enough effort into clean and coherent designs for management interfaces so that the average user could manage their own system.
  - The typical computer owner / user is no longer a technician or technologist. It is the responsibility of software designers and vendors to recognize this and provide systems that improve their chances of survival.
  - Safer default configurations.
  - Tutorial interfaces.

Could we be proactive? - 4

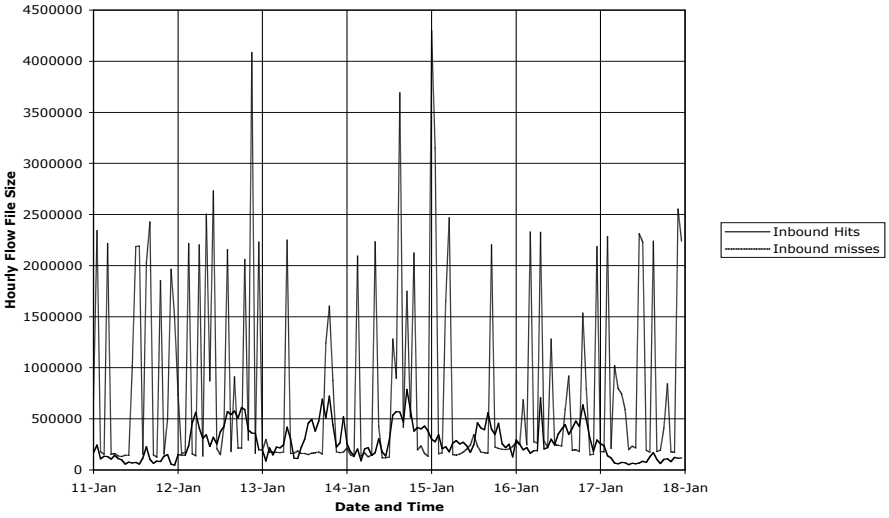
- Exploit developers are viewed as heroes in some circles.
  - I want a hacker's hall of shame.
  - *With heads on pikes.*
- Programmers who commit sins such as the creation of buffer overflows need re-education at best
  - Cut off their coding hand?

DALHOUSIE UNIVERSITY Inspiring Minds
Faculty of Computer Science
Privacy and Security Lab

If we have to react

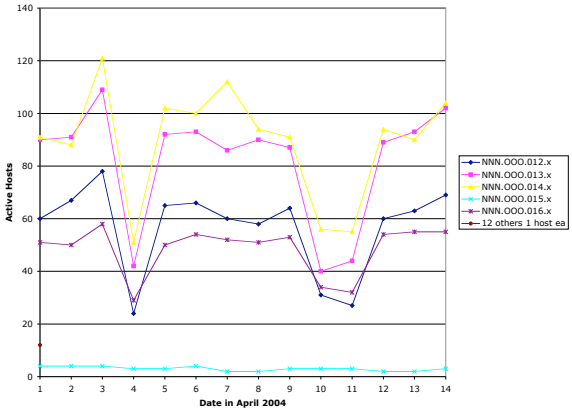
- Lets try to do it from a point of better information.
  - Comprehensive network observation (including, especially unused addresses)
  - Unbiased analyses (let the data talk)
  - Visualizations that provide insight.
- I'm going to conclude with a few more pictures:

One week on a /16



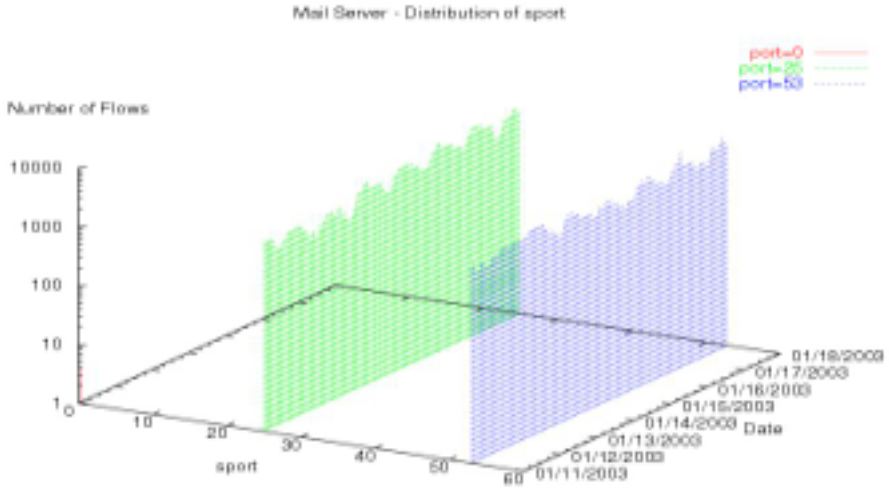
DALHOUSIE UNIVERSITY Inspiring Minds
Faculty of Computer Science
Privacy and Security Lab

NNN.OOO.x.x - Host activity



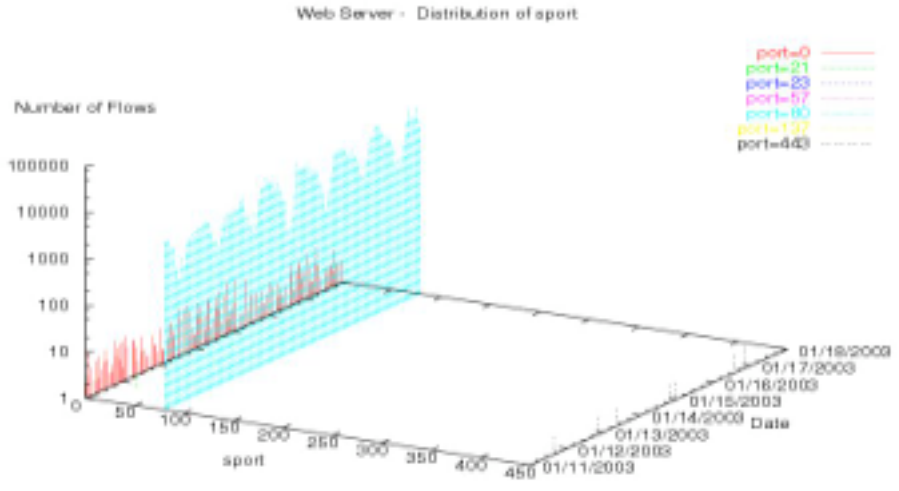
© 2005-2006 by John McHugh

Mail Server?

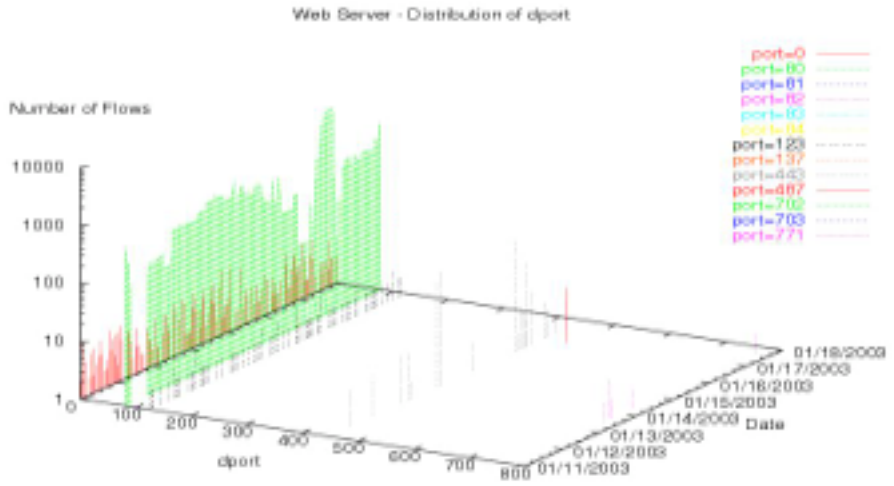


DALHOUSIE UNIVERSITY
Inspiring Minds
Faculty of Computer Science
Privacy and Security Lab

Web Server

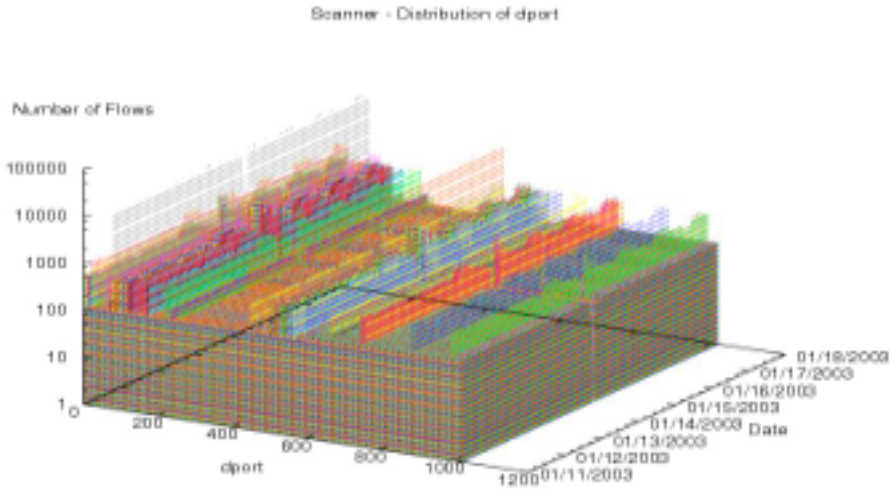


Web Server



DALHOUSIE UNIVERSITY
Inspiring Minds
Faculty of Computer Science
Privacy and Security Lab

Scanner



Thank You

- I'll be around for the rest of the conference.
- You can reach me as mchugh at cs.dal.ca
- Ideas for possible collaborations are welcome