Cohen and the First Computer Virus
What do we want to discuss today?

- Short biography of Fred Cohen
- Virus – The theoretical view
- Between ideas and reality
- Virus – Practical experiments
Short biography of Fred Cohen

Virus – The theoretical view

Between ideas and reality

Virus – Practical experiments
Short biography of Fred Cohen

- Professor of Computer Science / Electrical Engineering
  - 1985 – 1988
- One of the first virus researcher
  - Wrote several papers (1987, 1989…)
  - Did several proofs with Turing Machines
- Member of ACM, IACR, IEEE, etc.
Short biography of Fred Cohen

Today‘s activities

- Deception Toolkit (Linux)
  - Honeypot, created ~1998

- Security consulting service
  - Business inspections
  - Employee security training
Short biography of Fred Cohen

Today‘s activities

Also does

- Digital forensics
- Digital crime scene reconstruction
Short biography of Fred Cohen

- Virus research was complicated
  - No “real” virus existed “in the wild”
  - Nobody wants to have
    - “dangerous” experiments in their PC-environment
    - Encourage students to program a virus

- In his theoretical paper, all the helpers are only given by their first names!
  - “sensitive nature”
Virus – The theoretical view

- Short biography of Fred Cohen
- Virus – The theoretical view
- Between ideas and reality
- Virus – Practical experiments
“We define a computer ‘virus’ as a program that can ‘infect’ other programs by modifying them to include a possibly evolved copy of itself.”

- By F. Cohen, “Computer Viruses”, 1987
Virus – The theoretical view

- Infect -> spread through a computer or a network
- Every infected program also acts as virus
  - Exponential growth
    - But infected programs can’t be infected twice!
- Evolving ~ some kind of polymorphism
  - Virus detection is more complicated
Virus – The theoretical view

- An example virus

```plaintext
program virus :=
{1234567;
 subroutine infect-executable :=
 {loop: file = random-executable;
  if first-line-of-file = 1234567
   then goto loop;
  prepend virus to file;
 }
 subroutine do-damage :=
 {whatever damage is desired}
 subroutine trigger-pulled :=
 {return true on desired conditions}
 main-program :=
 {infect-executable;
  if trigger-pulled then do-damage;
  goto next;
 }
 next:}
```
Is a virus detection possible?

- The determination of: Given a program P, “Is P a virus?” is undecidable.
- Say there exists a decision procedure ‘D’, which decides ‘V’ is a virus, if ‘V’ infects another program.
- So virus-‘V’ is detected by ‘D’.
Virus – The theoretical view

- But now we modify ‘V’ to ’CV’
- ‘CV’ will not infect other programs, if ‘D’ decides, that ‘CV’ is a virus.
- If ‘D’ decides ‘CV’ is not a virus, than ‘CV’ will infect other programs.
- So ‘D’ is not the desired decision function
  - Because ‘D’ was an arbitrary function, this function does not exists.
So we can’t decide if a program is a virus or not.

Other proofs about viruses are done by Cohen using a Turing Machine.

We will now see one example
A Turing Machine has the following characteristics:

- A finite number of states
- A tape head
  - Moving is possible in different directions (-1;0;+1).
- A semi-infinite tape (only in one direction)
Virus – The theoretical view

Turing Machine

<table>
<thead>
<tr>
<th>S</th>
<th>I</th>
<th>N</th>
<th>O</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current State</td>
<td>Input (on Tape)</td>
<td>Next State</td>
<td>Will be written on tape</td>
<td>Tape Head Movement</td>
</tr>
</tbody>
</table>

Finite State Machine

Tape Head

Tape

Cell 0  Cell 1  Cell 2
We also use ‘macros’ here

- So our turing machine table can be shorter
  - I only show a short description of these ‘macros’.

C(0,1,2)

- Changes every occurrence of ‘0’ on the tape to ‘1’ until it reads the ‘2’ on the tape.
  - Moves right while doing this, next state is the state before the current state
Virus – The theoretical view

- **L(0)**
  - Moves left, until it reads the ‘0’ on the tape
    - Movement (-1), next state after reading ‘0’ is the state after the current state.

- **R(0)**
  - Moves right, until the ‘0’ occurs in front of the tape head
    - Movement (+1), next state after reading ‘0’ is the state after the current state.
Proof by demonstration...

<table>
<thead>
<tr>
<th>S</th>
<th>I</th>
<th>N</th>
<th>O</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_0$</td>
<td>L \text{ else} &amp; $S_1$ &amp; L &amp; +1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$S_0$   &amp; $S_0$ &amp; X &amp; 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_1$</td>
<td>0      &amp; C(0,x,R) &amp;   &amp;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_2$</td>
<td>R      &amp; $S_3$ &amp; R &amp; +1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_3$</td>
<td>&amp; $S_4$ &amp; L &amp; +1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_4$</td>
<td>&amp; $S_5$ &amp; X &amp; 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_5$</td>
<td>L(R)   &amp;   &amp;   &amp;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_6$</td>
<td>L(X or L) &amp;   &amp;   &amp;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_7$</td>
<td>L      &amp; $S_{11}$ &amp; L &amp; 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X      &amp; $S_8$ &amp; 0 &amp; +1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_8$</td>
<td>R(X)   &amp;   &amp;   &amp;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_9$</td>
<td>X      &amp; $S_{10}$ &amp; 0 &amp; +1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_{10}$</td>
<td>&amp; $S_5$ &amp; X &amp; 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_{11}$</td>
<td>R(X) &amp;   &amp;   &amp;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_{12}$</td>
<td>&amp; $S_{13}$ &amp; 0 &amp; +1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_{13}$</td>
<td>Wolfgang Apolinarski &amp; $S_{13}$ &amp; R &amp; 0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Virus – The theoretical view

So what does this Turing Machine do?

- Start with L
- Change 0 to X till R
- Read R, write R, +1
- Write L, +1 (S₄)
- Write X, L(R) (S₆)

L 0 R
L 0 R
L X R
L X R L
L X R L X
L X R L X

L X R L X
L 0 R L X
L 0 R L X
L 0 R L X
L 0 R L X
L 0 R L X

L(X), S₇
R(X), S₉
X, L(R), S₆
L(L or X), R(X)
S₁₂

Huh? S₁₃
Halt State

Wolfgang Apolinarски
Virus – The theoretical view

- So the L0R on the tape changed to L00R
  - So it is not a “simple” virus, it is polymorphic

- And we’ve shown another thing
  - If this virus would not have a halt state, but instead repeat his program, what would happen?
Virus – The theoretical view

- It would write infinite often the L0..0R phrase to the tape
  - Exactly: Countable infinite often

- Conclusion: There exist countable infinite viruses.

- But there also exist countable infinite number of different programs on a TM
  - So there exist as many viruses as programs!
Virus – The theoretical view
Summary

- A computer virus
  - Infects other programs
  - Can evolve (polymorphism)

- There exist as many viruses as programs on a computer
Between ideas and reality

- Short biography of Fred Cohen
- Virus – The theoretical view
- Between ideas and reality
- Virus – Practical experiments
Are there potential benefits of viruses?
   - Yes!

A compression virus which compresses binary files after infection
   - Could save over 50% of space normally taken by executables
     - In the eighties hard disk space was expensive!
This “virus” should ask the user for permission
- So it is no Trojan horse, but a virus!

Today many executable are already compressed
- So no need for a compression virus?
Between ideas and reality

- Benevolent viruses?
  - Cohen did write a paper about this in 1991
- Viruses for everything
  - Maintenance tasks
  - Garbage collection
  - etc.
- If one virus would fail, another would take his place
Between ideas and reality

- Man only needs to write a successor virus for a ‘program update’

- Distributed calculations with viruses?

- Failsafe database with virus support
  - A bill collector virus
Between ideas and reality

- So the whole database is distributed along the network
- No regular “scanning” for a bill is necessary
- The viruses awake by themselves and ‘learn’ when they have to be active

Artificial Life!
Between ideas and reality

Prevention of viruses?

- If sharing is allowed, a virus can spread to every user who takes part at the sharing
  - Virus paths are transitive!
- If modification of software is allowed, than a virus can reach new programs.

- Disallowing one of these?
  - Unacceptable, especially if teamwork is desired
Between ideas and reality

“Isolationism”
- Gameboy, other games consoles
  - Sharing is not allowed, but modification
    - Save games!
- Non-updatable firmware
  - DVD-Players, etc.
    - But most do have a flashable ROM!
Between ideas and reality

- So “Isolationism” is not a solution
- Some complicated security policies?
  - Unix file systems / NTFS partitions
    - Only slow down virus distribution, because not all users are affected

- New Idea: “Flow distance”
Between ideas and reality

Flow distance

- Special metric, that keeps tracks of the number of sharings, ie. the data flow
  - $\text{Max}(\text{distance(process)}, \text{dist(file)}) + 1$
    - If it is greater than a threshold, access is denied
  - But if all users have direct connections, this doesn’t help a lot.

- ‘Flow list’ lists all users that had effect on an object
Between ideas and reality

Flow distance

- Access is only granted, if a ‘trusted’ user has touched the object
- A metric is also possible:
  - Only access files where ≤ 2 users were involved
- Files of a distrusted user can be fully ignored

With this distance metrics a virus spread could be slowed down or stopped
Between ideas and reality
Between ideas and reality
Between ideas and reality

Summary

- Useful viruses
  - Distributed computing, compression virus
  - Artificial Life

- Prevention - “Isolationism”
  - Games console - firmware
  - Flow distance
Virus – Practical experiments

- Short biography of Fred Cohen
- Virus – The theoretical view
- Between ideas and reality
- Virus – Practical experiments
Virus – Practical experiments

- How to study the behaviour of a computer virus?
  - No virus existed in 1983
  - So instead of using an existing virus, a new one was written
- On the 3rd of November 1983 conceived
  - On the 10th presented
    - In a seminar on computer security
  - 8 hours of (expert) work
The virus infected a unix program called “vd” and spread using the system bulletin board
- No damage routine, only creates reports
- Traces to detect the virus everywhere

Five experiments took place
The attacker got all system rights in an average of 30 minutes!

Everybody was surprised about the short time, the virus had “success”

As result the administrators did not allow any other virus experiments to take place
Virus – Practical experiments

- So it was not intended to establish more security, but to “stay” at the current level
  - If no virus exists, no anti-virus actions had to be taken

- Other experiments were planned and viruses for different systems written
  - After several months the administration decided to not allow this experiments
    - The security officer even refused to read the proposals
Virus – Practical experiments

- So it was not allowed to add traces to the system, to discover a potential virus attack

- This reactions were typically for this time
  - Computer system were expensive so buying equipment only for virus testing was quite unrealistic
  - A “real world” scenario can’t take place in a sandbox
Virus – Practical experiments

In 1984 a virus on a system which used the Bell-LaPadula security policies was developed

- Bell-LaPadula allows a lower user not to read the higher users file. A higher user is not allowed to write in a lower users file
  - Security of information
  - System was in use by the US Air Force
Virus – Practical experiments

- The virus needed 20 seconds for each infection!
- After 18 hours the first infection was performed
- After 26 hours the virus was shown to administrators and programmers
  - It could cross all security boundaries, write down and read up…
Virus – Practical experiments

- On an unix system the infection was slowly, until it reaches a system administrator account
  - Especially “root”
Virus – Practical experiments
Results / Countermeasures

- Separate system administrator accounts and the normal user account
  - This separation was never really thought of.

- If a user announces a new program, one of the first users always was a system administrator…
  - Virus spreading is made very easy…
Virus – Practical experiments

- This discussion also applies to today’s computers
  - Windows – Vista’s new behaviour
He thought of developing an antibody for a virus

- Which also evolves by itself, in addition to human development

- He never used the term "Anti-Virus"
Virus – Practical experiments

Summary

- How to study virus behaviour?
  - Write an own virus
    - Study its behaviour
- Administrators & security personnel might not be helpful
  - Threats are everywhere ;-) 
- Viruses spread very fast, if a computer user uses his normal administrator account only