Seminar “Malware”
Prof. Dr. Joachim von zur Gathen, Daniel Loebenberger
WS 2007/2008

Cohen and the First Computer Virus

Author: Wolfgang Apolinarski
Date: 15.11.2007

Table of Contents

abstract:

Fred Cohen was one of the first computer virus researchers. He started with theoretical work in the nineteen-eighties, did several proofs about the appearance and the behaviour of viruses. His definition of computer virus is the first concrete definition about when to determinate, if a program is a virus. Artificial viral life and virus prevention are other topics he thought about, with results which are still important for today's virus research. He did the first virus experiments, against the objections of security personnel and system administrators, with the help of self programmed viruses. His findings encouraged computer scientists to think about establishing more security and to encourage people to act more responsible in the work with unknown programs and secret information.

Short Biography of Fred Cohen

Fred Cohen was a professor of Computer Science and Electrical Engineering at LeHigh University from 1985 till 1987 and, from 1987 till 1988 a professor of Electrical and Computer Engineering at the University of Cincinnati[1]. He is also a member of some well-known computer science orientated organizations like ACM (Association for Computing Machinery), IACR (International Association for Cryptologic Research) or IEEE (Institute of Electrical and Electronics Engineers).He was one of the first virus researchers and wrote several paper about viruses. He also explained that the name "virus" was invented by Len Adleman in 1983[2].

His research had to deal with several problems. Because there existed no real virus, he and his team had to program their own viruses, to show what abilities a virus has. Also there were some administrative problems. Since it was often not allowed to do some necessary experiments, he had to persuade several administrators and security personnel, because they feared a virus could brake down their expensive equipment. Cohen also was one of the first who did theoretical proofs of viral behaviour. He was also aware of the consequences for the people who helped him programming or setting up the experiments, so he only gave the first names in his paper, due to the "sensitive nature" of his research.[3]

He currently works with his company "Fred Cohen & Associates" on security related topics. These topics differ from his previous work on the universities and his previous theoretical research. He developed one of the first honeypots, the "Deception Toolkit", for tracing viral activities and to detect an intrusion in security systems. Such honeypots are nowadays in wide use for intrusion detection or to observe the threats, which a computer system is exposed to, e.g. in the internet. Cohen now has a security consulting services for enterprises. He does business inspections and employee security training. His company works out policies, e.g. for the right behaviour with passwords and looks for holes in the current security concept. They also visit companies and check there current equipment for weaknesses or develop a new security architecture. This work is normally well-paid by the companies and shows that Cohen's engagement about virus research is versatile. Today he also works with viruses and other security threats, but now from a more practically point of view. So he uses his theoretic papers as basic principle for his practical virus defender job he has today. Another part of his work is digital forensics, where he does data reconstruction and sometimes is a consultant in a lawsuit.[4]

So he uses his theoretical knowledge to solve problems occurring in the real world. He changed from being a mathematician/computer science researcher to a consultant of profit orientated companies.

1 cf. [Coh89] p. 344
2 cf. [Coh87] p. 31
3 cf. [Coh87] p. 35
4 cf. [FCA07] http://all.net/forsale/Services.html

Virus – The Theoretical View

At first we introduce a definition made by Fred Cohen, which defines the term “computer virus”:
“We define a computer 'virus' as a program that can 'infect' other programs by modifying them to include a possibly evolved copy of itself.”[5]
So it can easily be seen, that his definition has several different aspects on which we could have a closer look. At first the virus program must have the 'infect' ability. It changes other programs, so that they include a copy of the virus program. This copy simply could be a real copy of a virus which is just prepended to a given program, so it is called before the original program is called and this program is called afterwards. That means infection through modification of the original program. Another point of this definition is the “evolved copy”. This is not a must, so a virus does not need to evolve its copy, but it is possible. Today polymorphic viruses “evolve” with every copy (after a specific algorithm, to complicate the detection from an anti virus scanner). But a real evolution, like artificial life, does none of the current viruses have. So Cohen's definition of a computer 'virus' also includes useful programs, e.g. a compiler who compiles another version of itself. It includes an evolved copy of itself in another program. Nevertheless, such useful “viruses” are not the perspective of current virus research, since a semantic decision about:”Is program A a virus” is not easily done (I just remind me of somebody, who installed a trojan horse on one of his PCs to remote control it from another room), neither is a syntactic, as some heuristics of current anti virus programs proof (false positive). Also this definition could allow some thoughts about the infection speed. Since every virus will infect at least one program, when it is called, one can be sure, that, the infection spreads very fast, because every infected program also acts as a virus. This leads to an exponential growth of viral infections. This growth will stop, when there are not enough virus-free programs available to spread to. In big computer networks, like the internet, this could need some time, but at a single workstation this is reached quite fast. So the exponential growth has an upper bound, which is near the approx. half of infectable programs. Then the number of infections still grows, but at a lower speed as before (because it is not as easy as before to find an infected program). If the virus “must” infect a new program every time it is called, it will need an amount of time, so an infection could be discovered very easy, but its infection speed will behave different then described before, because every call really infects a new program, because this virus could be easier discovered, I don't discuss this virus specifically.

Cohen also provided an example virus, which is shown on the right[6]. This example virus introduces two other features. It has a trigger and a damage doing subroutine. If we look at the main-program, we see, that this virus first infects a new random-executable. It will loop around to find files which are not infected (A file will be seen as infected, if it starts with the digits “1234567”, this is the “label” of this virus). If has found one, it will infect it, by prepending itself to the file. Subsequently it looks if the trigger was pulled (more details follow) and does damage (see above). Then the original program, which was infected, is called (this is the mark “next” on the bottom of the image, after this mark the normal program should follow, which is left out on the picture).

```
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:}
```

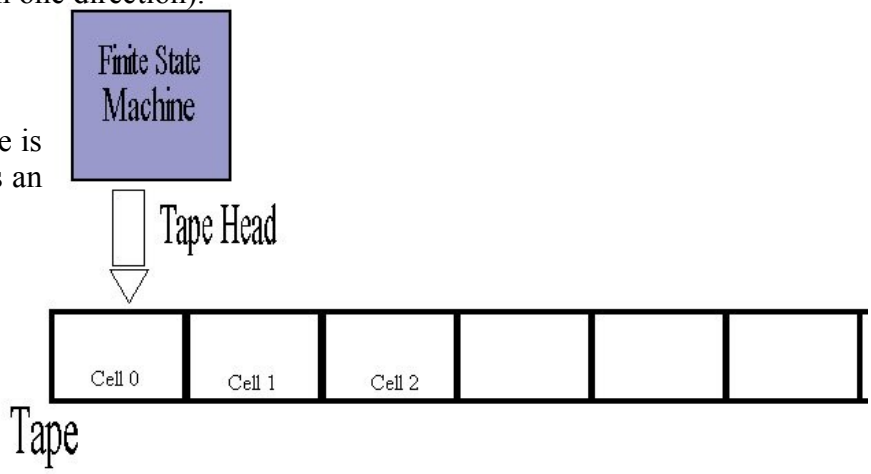
The trigger could be anything we think of. Maybe a concrete date. So we send out our virus and on a specific date, this virus will start to do damage, perhaps delete all our Word-Documents. A virus is

5 [Coh87], p. 23
6 Image from [Coh87], p. 23

often firstly discovered, if it has done some damage. Then this is too late, because we possibly spread our virus to the whole network and it would delete all important files at once (on the specific date). This “trigger” was already used by many viruses such the “Michelangelo” virus or other modern worms, who try to spread during the first part of the month and try to attack a computer if the second part of the month has begun.
We now concentrate on another question of theoretical computer science:
“Is a virus detection possible?” We will see, that this question is undecidable:
Say there exists a decision procedure 'D', which decides if 'V' is a virus. So the current virus-'V' is detected by 'D'. But if we now modify the 'V' to a new program 'CV' which invokes the decision procedure 'D', we will see, that 'D' is contradicted. 'CV' will first invoke 'D' and if 'D' decides 'CV' is not a virus, it will infect another program. If 'D' decides, that 'CV' is a virus, 'CV' will not infect other programs (and so not act as a virus). So 'D' is not our desired decision function. But because 'D' was an arbitrary function, we could see, that such a function does not exists. So the determination of is program 'P' a virus is undecidable[7]. But this is not new to users of current anti virus software, they would recognize viruses by viral behaviour (and do not use such a decision function). So the above shown example virus is easily detected, because of the “1234567” on the beginning of every infected file. But not all potential viruses “detected” by anti virus are real viruses (Think of the EICAR test virus or heuristics).
So we can't decide if a program is a virus or not. Cohen did also some other useful theoretical virus research. Many of his proves use the so-called Turing Machines.
A Turing Machine has the following characteristics:

- A finite number of states
- A tape head
  - Which can move in different directions, to the left (-1), to the right(+1) or stay at the current position (0).
- A semi-infinite tape (infinite long in one direction).

Normally the program of a Turing Machine is given as a table. The table shown below is an example table with detailed description.



S	I	N	O	D
Current State	Input (current position on the tape)	Next State	Will be written on the tape (on the current position)	Tape Head Movement (+1;0; -1)

Cohen uses three abbreviations to a normal Turing Machine table. These are also given as a table, so they do not change the specifications of a Turing Machine. I will now give a short description of these abbreviations, the tables are also shown below.
With L(X), the tape head moves in the left direction, until it reads the “X” on the tape (i.e. L(X or

7 cf. [Coh87], p. 28, 4.1

O) means, that the tape head moves left, until it reads “X” or “O” on the tape) and stopps at this position. R(X) is the same as L(X), but for the opposite direction, so the tape head moves right, until it reads the “X”. Another macro, C(X,Y,Z) has three arguments. It means change the “X”s to “Y”s until you reach the “Z”[8]. So all “X”s on the tape are changed to “Y”s (the tape head moves right, while doing this) and if there is a “Z” on the tape (Input “I” in the table above), than the macro will stop. After one of the macros (L(X), R(X) or C(X,Y,Z)) ran, the next state in the table will be executed.

R(X)[9]:

S	I	N	O	D
S_n	X else	S_{n+1} S_n	X else	0 +1

L(X)[10]:

S	I	N	O	D
S_n	X else	S_{n+1} S_n	X else	0 -1

C(X,Y,Z)[11]:

S	I	N	O	D
S_n	Z X else	S_{n+1} S_n Sn	Z Y else	0 +1 +1

The Turing Machine program will now proof, that there are countable infinite many viruses. This will be shown using a “polymorphic” virus “L0R”, which changes to “L00R” or “L000R”, “L0...0R”, etc. The program table is in the appendix.

So this Turing Machine will change a “L0R” virus on the tape (starting with the tape head on the “L”) to a “L0RL00R”. If the Turing Machine would not halt at the end of the table-program, the virus would be written countable infinite often on the tape. Since our today's computers are Turing equivalent (they don't have an infinite tape, but all the calculations done with a Turing Machine can also be done with a modern computer (and backwards)), there exists countable infinite many viruses. But because there also exist countable infinite many programs (not proved here), there exists as many programs as viruses (the cardinality of viruses is equal to the cardinality of programs).

8 cf. [Coh89], p. 335
9 from [Coh89], p. 335
10 from [Coh89], p. 335, but: Following state is always the state after the current state
11 from [Coh89], p. 335, but: Following state is always the state after the current state

Cohen also thought about “Benevolent computer viruses”[12]. He did this, because of the computational potential, that distributed computing has. A virus did “384 Billion operations per second”[13] and this would be really useful, if one could use this for a good case (and not for doing damage/infecting other programs, as normal viruses do). One of his easier thoughts is a virus, which compresses executables and prepends itself and a decompression program in front of the “infected” program[14]. This virus would have been really useful in the nineteen-eighties, because the hard disc space was very expensive and so every user only had little space to store his programs and documents. He calculated, that this virus could save up to 50% of the disc space, which normally was taken by executables. If we look what happened, we will discover, that today most programs are already compressed by their manufacturer. So there might be no need for a special compression virus anymore. But his thoughts go many further. He thought of viruses, which could be active during the time the computer is idling. Many computers are the most time, that they are active, just waiting for input of the user or they have a high performance which is not used most of the time, especially if the user is only using his electronic office and does not play sophisticated modern computer games. So this viruses would only use the computer, when the CPU is in idle mode, so a user would not be disturbed and the computer is not a good looking heating. This sounds like a distributed computer project like Seti@Home[15] or like the Mersenne prime search project[16]. But because he doesn't want to use normal clients, this is different. He wants a “virus”, which evolves itself and not during manual software updates. He introduces a “bill collector” virus[17] which evolves during collecting bills. The new evolved version includes the collected bill and sets itself in sleeping mode, after it made some flags in the scheduler table, when it should awake again. Because this concept also uses “children” and a “gene pool”, Cohen's ideas also touch the field of Artificial Life. A database could so be distributed through the whole network, which would not lead to a single point of failure anymore. His ideas to maintenance viruses[18] include viruses, that control each other like a real ecosystem. So there are viruses which awake viruses, that slept to long and other viruses can kill a virus process, if it hangs in a loop.

If we think about viral behaviour, we could also think of prevention mechanisms to prevent virus spreading. Cohen did some basic work on this point. If it is allowed for users to share their data files, than a virus can spread to every user who takes part at the sharing[19]. It can also spread to a user, who does not take part at the sharing, but shares data with one of the user who takes part. So the paths for virus spreading are transitive. If user A shares files with user B, and user B needs some files from user C, also user A gets the virus, if user C is infected. Another point is, that if we don't allow modifications of programs, a virus can not spread. Almost all computers allow modifications in their random access memory, so a virus could spread using only this, but if the computer is rebooted, the virus vanished (as every content of the RAM vanishes, if the computer does a reboot). So, if we disallow modification of software (i.e. Data) or disallow sharing, we would not have virus infections spreading through whole networks (except for temporary viruses, because of the RAM). But disallowing modifications or sharing is unacceptable, because most work which should be done with a computer needs modifications and sharing is necessary, especially, if teamwork is desired, otherwise one only could work with self-programmed programs, which could be useful in a special

12 cf. [Coh91]
13 [Coh91] “Background”
14 cf. [Coh87], p. 24
15 http://setiathome.ssl.berkeley.edu
16 http://www.mersenne.org, GIMPS
17 cf. [Coh91], “The Viral Bill Collector” (here simplified)
18 cf. [Coh91], “Maintenance Viruses and the Birth/Death Process”
19 cf. [Coh87], p. 25

case, but normally isn't something we want to achieve.
But this “isolationism”[20] does already exist in real computer system. E.g. the Game Boy and other games consoles are isolated, because they have a game (read only memory) which is isolated. If the game allows save games, than this normally is not a point where a virus could infect the games console, because save games are not executed and only loaded. So sharing is limited, and virus spreading for a games console is not something somebody ever heard about, but with the new more computer like games console (Playstation 3 can run Linux), this could be a new thread, because it would give up the isolation concept. Another computer system are firmwares for different devices. Like a stand-alone DVD-player or a VCR, but many of these firmwares have a flashable ROM, so one could think of a virus for a DVD-player, but because spreading on read-only memory (DVD-ROM) is complicated, this is not a current threat. Only on personal computers, who could more easily get infected, a DVD-Player could be harmed as side-effect of a computer virus.
So “isolationism” is not a solution, but we could think of other security policies. Modern file systems, like ext3 or NTFS allow different read/write/execute access rights for files. So if one user executes a virus, only his user-space is infected and only user who share with this user are at risk to get infected, too. Only if the root (or another system administrator) gets infected, all users on the system are also infected. But practical experiments have shown, that this policies do not prevent virus spreading, but only slow down the virus infection (which could also be important, since it is possible to analyse the virus and to create a counter strategy to prevent further spreading). Cohen thought of a flow distance for every process and data file[21]. This is a special metric, that traces the data flow. So it uses the formula: $\max(\text{distance(process)}, \text{distance(file)})+1$ for every new file, that a running process on a machine wants to open. So if we have only peer-to-peer connections, this could help to stop the spreading of the virus (at the cost of more computing power used for maintaining the flow distance). So if user A has a connection to user B, and user B has a connection to user C. Then every file from user A, which is used by a local process (distance 0) on the system of user C has the distance 2. If now user V has a virus and is connected to user A, and all users reject files, that are from the distance greater than 2, the virus could spread to user A (distance 1) and user B (distance 2), but not to user C (distance 3). Even if it would write files to the computer of user B, this is not possible, because all files written by user B with the help of a process or file of user C would have a distance of 3 (So a local process from user C (distance 0) has to change its state to distance 2, if it opens a file from user A). But because today many networks have direct connections, so this would not help against virus spreading.
Another idea is to implement a “Flow list”[22], which logs all users, that had access on a specific file/object. So one could apply a policy that all files opened by a user first have to be touched by the trusted user T. This could help to reduce the speed of a virus spreading efficiently. Also files of distrusted users could be fully ignored. In current operating systems, some kind of this flow model is implemented. Files that are downloaded from the network have to acknowledged from the user, before they are executed, if they don't have a digital certificate (from a trusted authority). Also the flow distance model is introduced, every file, that comes from a different computer (over the network) is marked by the operating system and it is asked if you want to execute this file (so you recognize that this file is from a foreign network).
So Cohen's ideas are partly realized, but especially the artificial life part is not ready for the market, yet.

20 cf. [Coh87], p. 25
21 cf. [Coh87], p. 26 3.3 “Flow Models”
22 cf. [Coh87], p. 27

Virus – Practical Experiments

As already said in chapter 1, it is not easy to study the behaviour of a computer virus, if there does not exist one (Except some "strange", wrong behaving computer programs like the "Xerox worm"[23]). Thus Cohen and his team had to write their own virus and it was presented on the 10th November of 1983.[24] It took eight hours of expert work to construct this virus, afterwards they performed some experiments in Unix-like environment. The virus infected a unix program called "vd", where Unix structures are displayed graphically,[25] and had no damage routine it only created reports and made traces to detect the virus everywhere, to prevent uncontrolled spreading. Cohen did five similar experiments take place, where some of the users were informed, that such a virus experiment will take place. The result was, that the attacker virus got all system rights in an average time of 30 minutes. Also the users who were informed, that a virus attack will take place got infected. This short time the virus needed was very surprisingly to every observer of the experiment. As result, the administrators did not allow any other virus experiments to take place. This was very disappointing for Cohen and his working group, since they want to establish more security even for a potential "new" viral attack. But the administrators wants to "stay" at the current level of security, so if no virus exists, no anti-virus actions had to be done. Such behaviour is also known from today, many administrators think, that they better protect their equipment (which was really expensive in the nineteen-eighties), if they don't allow dangerous experiments. But security can not improve, if nobody knows how a virus behaves. So Cohen planned more experiments and his team wrote many viruses for different systems, they offered the security personnel to observer all experiments, so they could learn from the behaviour of the virus, but after several months the administrators decided to not allow this experiments. One of the security officers even refused to read the proposals.[26] But because a "real world" scenario can't take place in a sandbox and simulators and virtual machines were not that powerful as they are today, Cohen did not give up and his team wrote a virus which was designed to circumvent a security policy system. The Bell-LaPadula system secures the information. Lower users are not allowed to read files of higher users (no read-up). Higher users are not allowed to write in a lower users file (no write down). Such systems are often in use by government agencies or other enterprises who need the security of information. The virus was programmed by a programmer, who was not experienced with the system, so the virus needed 20 seconds for the infection step (Reminder: copy yourself in front of the infected file). They marooned the virus and 18 hours later, the first infection was performed. 8 hours later, they could present the virus to administrators and programmers. It could cross all security boundaries, so write down and read up[27]. So the Bell-LaPadula system was compromised. Afterwards Cohen made a review of his experiments. The outcome was, that a infection on a unix system was slowly, until it reached the account of a system administrator, especially the "root" account. Thus they proposed to separate the account of the system administrator from the normal user account, i.e. a system administrator who also uses the system for his daily work should have a separate user account[28]. This separation was very unusual, because there were no virus threats in that time. Also they discovered, that one of the first users of a newly announced program was always a system administrator. So virus spreading was made very easy.

On today's computers, this discussion applies also. Many unix-like environment separate the accounts of the system administrator and the user. It is possible to change the simple user rights to administrator rights, if it is needed via a system command. Also "Windows Vista" introduced a new

23 [Coh87], p. 22
24 cf. [Coh87], p. 31 "The First Virus"
25 cf. [Coh87], p. 31
26 cf. [Coh87], p. 32
27 cf. [Coh87], p. 32 "A Bell-LaPadula Based System"
28 cf. [Coh87], p. 33 "Summary and Conclusions"

feature, the User Account Control (UAC), so every administrator has only user rights. He has to approve the use of his administrative rights on the "secure desktop".[29] So viral spreading is bound to the user account and strongly slowed down. But real protection from the virus is only achievable with isolation.
Cohen's ideas of anti-virus was different than current anti-virus programs. He thought of antibodies which behave like a virus, but remove them. Again this is a reference to Artificial Life.

List of References

- [Coh87] F. Cohen: "Computer Viruses - Theory and Experiments", 1987, Computers & Security 6 page 22-35, Elsevier Science Publishers B.V. (North-Holland).
  Also available in a slightly different version from 1984:
  http://all.net/books/virus/index.html
- [Coh89] F. Cohen: "Computational Aspects of Computer Viruses", 1989, Computers & Security 8 page 325-344, Elsevier Science Publishers Ltd.
- [Coh91] F. Cohen: "A Case for Benevolent Viruses", 1991, DPMA, IEEE, ACM Computer Virus and Security Conference, March 1992
  Online available at: http://www.all.net/books/integ/goodvcase.html
- [FCA07] Fred Cohen & Associates, http://all.net/forsale/Services.html (15.11.2007)
- [MST07] M. Russinovich: "Inside Windows Vista User Account Control", June 2007, TechNet Magazine
  Online available at (in german, if the loc=en is exchanged with loc=de):
  http://www.microsoft.com/technet/technetmag/issues/2007/06/uac/default.aspx?loc=en

29 [MST07], directly under "Figure 13 Elevation Flow"

Appendix

Turing Machine which proves that there exist countable infinite viruses[30].

S	I	N	O	D
S_0	L else	S_1 S_0	L X	+1 0
S_1	0	C(0,X,R)		
S_2	R	S_3	R	+1
S_3		S_4	L	+1
S_4		S_5	X	0
S_5	L(R)			
S_6	L(X or L)			
S_7	L X	S_{11} S_8	L 0	0 +1
S_8	R(X)			
S_9	X	S_{10}	0	+1
S_{10}		S_5	X	0
S_{11}	R(X)			
S_{12}		S_{13}	0	+1
S_{13}		S_{13}	R	0

30 from [Coh89], p. 337