Paul Cockshott's Blog

Comments on economics and politics

TECHNOLOGY

Defence against Vault 7 attacks

Date: March 13, 2017Author: Paul Cockshott 0 Comments

In the past cyber attacks were largely carried out by amateurs or criminal gangs. Recent releases of information from Wikileaks have made it clear that a whole range of digital devices have been subverted by intelligence agencies.

More recently they have been commercialised with a market existing in the development and detection of 'exploits' or weaknesses. Firms advertise on their websites that they have hacks that allow them to control computers running web browsers such as the latest versions of Internet Explorer or Google Chrome.

Companies that develop these exploits then market them to NATO approved defence and intelligence agencies. In the light of the Snowden revelations it is reasonable to fear that

some of the Endpoint attacks exploits in widely used products originating from big US companies like Microsoft, Apple or Google may have been developed on the basis of inside knowledge. Either they are deliberately introduced into the software, or information about the weaknesses may leak to the exploits contractors prior to their being fixed in a subsequent software release.

Government agencies like the UK GCHQ and the US NSA systematically tap communications lines and hack into Internet switches to divert data.

Historic examples

There is of course a long history of deliberate interception of digital information by intelligence agencies. In the early 20th century the British government operated the 'red net', the network of British owned undersea cables connecting the world. It was labeled 'red' as the British Empire was marked red on the map. As telegraph and telex information passed through British territories they were systematically copied and decrypted by the intelligence service.

This control of data flows was of decisive importance to British strategic objectives. The first act of hostilities in the First World War was the dredging up and cutting of German controlled sub-sea cables. This occurred with hours of war breaking out. In consequence German diplomatic correspondence had to be encrypted and sent via neutral Holland and then through London to the rest of the world. Using this the British were able to obtain the text of the proposed secret treaty between Germany and Mexico to divide American territory in the even of war. Publication of this treaty in the American newspapers led to the USA declaring war in Germany in 1917.

Later the leaking of secret Commintern telegrams suggesting that the Commintern should support the Labour Party, were used to precipitate the fall of the first Labour Government.

Britain remains a hub for undersea cables, now fibre optic rather than electrical.

Edward Snowden has revealed that all internet communications passing from Europe to North America and on from there are systematically tapped by GCHQ. The revelation that German and other EU communications are still tapped by the UK government caused some controversy when it was revealed by Snowden.

Edward Snowden revealed that within the USA data centers and Internet hubs are linked to the NSA which carries out diversion of information flows similar to GCHQ.

The precautionary principle suggests that chat services like Facebook, Google and Skype can be assumed to be tapped at the premises of the companies running these services.

It is perhaps significant that after Microsoft took over Skype it switched the system to a server based rather than peer to peer based making it easier to tap. Peer to peer traffic does not go through a single server making tapping harder.

From the documents released by Wikileaks it has become clear that the actual encryption

algorithms are still secure, but the endpoints are easily compromised. This is one of the motivations behind our proposal: if both endpoints engaging in encrypted communication are secure, it is currently impossible to decrypt the communication.

Traditional viruses

These install themselves and then propagate copies either by email or external storage media contact (USB sticks, SD cards, ebook readers etc). Once installed, what they do is up to the writer of the virus.

The most sinister recent example was the Stuxnet virus, which damaged the equipment in the Iranian gas centrifuge plant. This showed evidence of having been expertly and deliberately constructed in order to produce damage, but not to the computer on which it was run, but to industrial equipment controlled by the computer.

- The virus was transmitted via USB sticks, copying itself onto any USB sticks inserted in the machine, and copying itself from any USB stick into all machines that the stick was placed in.
- It specifically targeted a particular type of industrial control software. If the computer it infected had no industrial control equipment it was benign.
- If it found the equipment, the virus made equipment operate outside safe parameters. It was particularly targeted at centrifuges processing uranium hexaflouride, which were made to run slightly faster than they should, so that they would rapidly wear out.

Although it can not be proven, it is generally assumed that the virus was developed by either Israeli or US stage agencies as part of an attempt to sabotage the Iranian nuclear power programme. It is viewed by commentators as part of an ongoing covert operation that has also involved targeted assassinations of Iranian physicists.

Buffer overflow attacks

These rely on badly written programmes running on computer A that allow messages sent from computer B to overwrite part of the subroutine linkage area in the data memory of computer A.

- When this happens the overwriting code gains control over computer A and can install malware. The malware may be activated at some later time to produce effects similar to those due to a virus.
- Such attacks can be launched from malicious websites.
- Secuirty exploits in discovered popular web browsers or display programmes such as Adobe Acrobat can take this form.
- It is important to note that a buffer overflow is a weakness even without injection of code: a buffer overflows can crash e.g. an ssh server, thus effectively resulting in Denial of Service.

Homeric Attacks

These rely on software that is apparently benign and useful, but which may contain secret malware. They are modelled on Homer's myth of the giant horse built by the Greeks as a gift to the King of Troy. Typically the malware then copies confidential information from the infected computer to computers controlled by the producer of the malware.

- The most notorious source of these are freely down-loadable utilities or apparent utilities that the user is induced to deliberately run on their computer.
- But should government A be able to obtain collaboration from major software companies, it would be possible to introduce Trojan horse software into widely used utilities such as Email readers, word processors, PDF file readers, spreadheets etc. If these were then used by countries B and C in their government offices, then government A might be privy to the confidential reports of governments B and C.
- In principle, it is possible to guard against this type of attack by stopping the process from communicating over the network, and many firewall products have this ability. However, in practice this is difficult because many applications exchange information with their software company to check for updates, and by blocking the communication, one stops vulnerabilities from being fixed.
- With our proposed approach however, the need for updates largely disappears as there is much less scope for exploiting vulnerabilities.

Targets

Telephone or internet switching points. These systems fail more gracefully than power networks, but a sufficiently widespread virus attack on switching points could cause significant interruptions to traffic. Air traffic control systems

The use of malware for espionage is an obvious danger, but one has to assume that the Stuxnet virus is just one currently known exemplar of a range of similar designs that have either already been, or could soon be, developed by state cyberwar departments. These would have a wide range of possible infrastructure targets:

- Modern atomic power stations [1]. The Chernobyl accident showed that operation of nuclear power stations outside their design parameters may lead to significant adverse effects.
- National power grids. Experience in the USA has shown that accidental overloading of a few switching points can lead to cascading failure: safety cut-outs come into operation redirecting current, causing more overloading and further cut-outs. Such cutouts induced by single accident points have led to blackouts lasting hours over substantial parts of the USA and Canada.
- Telephone or internet switching points. These systems fail more gracefully than power networks, but a sufficiently widespread virus attack on switching points could cause significant interruptions to traffic.
- Air traffic control systems and air defence radar systems, insofar as these have been upgraded to use modern commodity brand PCs. The hazards here are obvious.

The term ‘computer virus’ is a borrowing from medical terminology. Indeed the very idea of creating computer viruses came from a deliberate copying of biological ideas. It is not surprising that the first technical fix: anti-virus software, also borrowed from biology.

This works on the model of the vertebrate acquired immune system.

Our immune system learns to recognise pathogens and then produces antibodies to them. Our immune system learns to recognise specific amino acid sequences, or motifs, in proteins as belonging to hostile organisms, and produces anti-bodies which bind to and neutralise these proteins.

On first encounter with a new virus we have no defence (SARS, Ebola etc). Whole races can be wiped out on contact with unfamiliar new viruses: experience of New World and island tribes on exposure to Old World viruses like colds, flu, smallpox.

Likewise, antivirus software relies on the providers of the software recognising motifs in the malicious code and thus identifying it. A software motif is a sequence of bytes invariably found in the code of a particular virus.

A brand new virus will not be detected unless it shares motifs with previous versions.

Monocultures are vulnerable

Healthy	Blighted

Figure Monocultures are vulnerable

All organisms in a monoculture have similar genetic structure if a virus can infect one it can infect all. This is why food crops, where a single species dominates, are particularly vulnerable to infection.

A natural grassland will contain dozens of species, and the ecosystem as a whole is robust against the attack of individual microbial pathogens. A giant field of corn is vulnerable.

Computer system monocultures

Windows PCs and Android mobile phones are two examples computer system monocultures. The equivalent to the Genome of plants in the cyber environment is the machine code of the microprocessors.

Microprocessors with the same machine code and same operating software can be infected by the same malware. But binary malware for machine code X will not infect a computer with machine code Y.

An machine code is a list of numbers with special meaning to the computer for example

code	meaning
0	Load
1	Store
2	ADD
3	SUBTRACT
4	JUMP
…	

Machines can typically recognise hundreds of such codes. Android recognises 256. All programmes run on the system rely on having a standard interpretation of these codes. The meaning of these codes is normally fixed by the electrical circuit of the computer.

In the early days of microprocessors there were many competing code schemes from different manufacturers: Intel, Motorola, Hitatchi, National Semiconductor, Zilog etc all had their own proprietary codes. The process of concentration and monopolisation in the computer industry means that now that computers, tablets and mobile phones nearly all use just two designs of machine code: the US Intel code and the code developed by the UK ARM company. Each of these software companies has licensed their codes to chip manufacturers in other countries, who in turn sell the chips to electronics manufacturers producing final consumer and industrial products.

- Windows, Apple and Linux computers all use the Intel machine code.
- Nearly all smartphones and tablets use the ARM machine code.

Virtual machine codes

Not all programmes depend directly on the raw machine code. Some use a virtual machine code that is interpreted by software rather than hardware. Some widely used virtual machine codes are those from the following US companies :

Oracle:
the JVM into which Java programmes are translated.
Google:
the Dalvik code that is used for Android apps.
Adobe:
the Postscript and subsequent PDF codes used in printers and document viewers.
Microsoft:
the Macro code used in Word documents and Excel spreadsheets.

Whilst the existence of many hardware machine codes protects the computer ecosystem against virus spread, the proliferation of popular virtual codes makes it easier for viruses to spread.

This is because a single computer can now be infected by viruses adapted to several different codes. An Intel PC with virtual machines from Oracle, Microsoft and Adobe

installed provides 4 different targets for viruses to infect.

Permuted systems

The basic approach to protecting computer ecosystems should be to create a diversity of machine codes. But this runs up against the high cost of designing computers. A country could not afford to design and build hundreds of completely different microprocessor architectures. The contemporary cheapness of computers is due to millions of identical microprocessors produced by what amounts to a sophisticated printing technology.

We propose to retain this manufacturing advantage, but to introduce as part of the hardware a permutation unit that shuffles the machine code just before it is executed.

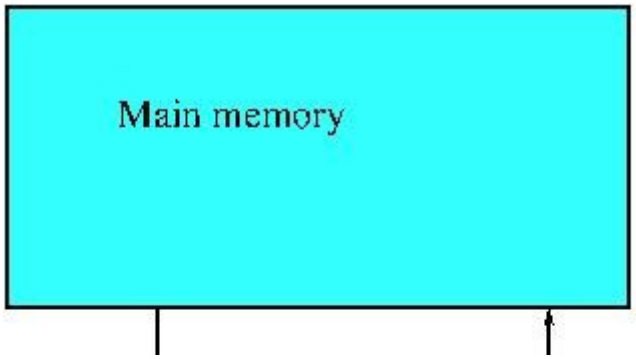
Permuting means re-ordering or shuffling things.

If you construct an chip device with a permuted set of meanings for its machine code no software designed for a standard chip will run on it.

code	meaning	Permuted meaning
0	Load	JUMP
1	Store	ADD
2	ADD	Load
3	SUBTRACT	Store
4	JUMP	SUBTRACT
…		

Thus no malware designed for the standard chip machine code will run on it.

It is in principle relatively easy to modify the design of a processor chip so that it incorporates a permutation unit that permutes the machine code. A sensible place to put the permutation unit is between main memory and the instruction cache, meaning that the circuit delay for permutation is only met on loading the cache rather than each time an instruction is executed.



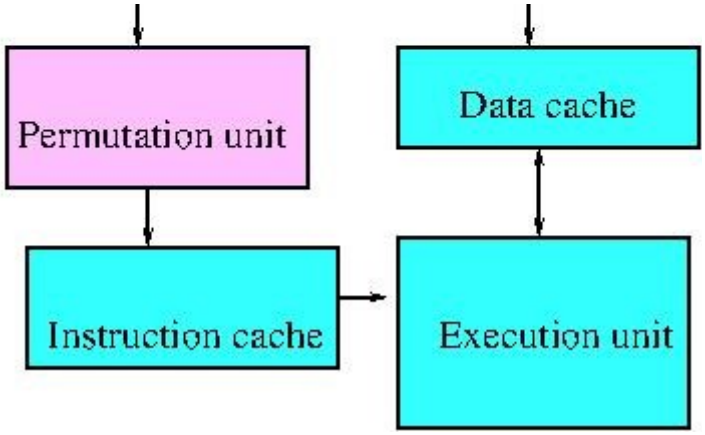


Figure Permute unit

Suppose you have a machine whose 32 bit codewords looked like this:

field	Opcode	r1	r2	offset
bits	8	4	4	16

One could use two alternative permutation techniques.

1. The simplest is to permute the order of the meanings of the opcode field as shown in our previous example. The machine has 256 possible opcodes, so a 256 element permutation table would be needed. This allows 256! different possible unique designs of machine code.
2. Alternatively one could permute or shuffle the bits in the whole instruction words. Since there are 32 bits, this would allow 32! factorial different codes.

Clearly the first alternative is much better. Each chip would have its unique permutation table stored in non-volatile memory, allowing the permutations to be loaded as a final manufacturing step on chips that were otherwise architecturally identical, and thus run unique machine codes on each of chips coming off the production line. One might allow a facility of field programmability of the permutation table.

Suppose that such chips were available and incorporated into PCs, tablets etc. For markets where there was no particular security problem one could use devices with a null permutation. These would run the native, unpermuted machine code supported by the hardware, but would run the risk of malware infestation.

Suppose the governments of countries B and C want to secure their government ecosystem against viruses, what do they need to do?

1. Assume that they each settle upon a specific version of Linux for their machines. Country B uses Debian and Country C Ubuntu. Each country has a server for distributing approved copies of Linux to government machines.
2. Each country maintains a secret database that associates with a identifier of every computer it uses a unique permutation table. The unique id can be conveniently calculated as a hash function of the permutation table.

3. When the government in country C buys a new computer it sends the machine to a depot containing $Server_C$. Staff at this depot construct two files: a permutation table, and a boot disk image containing a permuted copy of Linux.
4. The permutation table is burned into a ROM on the machine and the boot disk installed in the machine.

A binary to binary translator on the server would convert the raw public Linux software to a permuted version able run on a machine with a particular permuted instructionset. A similar binary translation process would be used to provide periodic software upgrades. In this case it would be necessary for the client machine to send its unique identifier when requesting a software upgrade.

Summary

- 'Species barriers' prevent the spread of viruses among the machines of either country B or country C.
- Buffer overflow attacks launched from country A against either B or C will fail as the introduced malware would not be a valid programme and so would cause an immediate crash on visiting the malicious website in country A. To guard against buffer overflow attacks that could bring down the entire system, the communication stack will be implemented in hardware.
- Homeric attacks would only be possible if malware was introduced into the source code of the Linux images used by the servers in B and C. The governments would have to insist that software was only mounted on the servers whose source code was:
  - Openly available
  - Had been subjected to inspection to attempt to detect obfuscated malware. This route could not be completely closed, but the ease by which such homeric attacks could be carried out would be much less.
  - Did not contain interpretive code. This is harder to ensure since so many contemporary Linux utilities depend on interpretive code. The consequence would be that the range of software supported on these machines would be strictly limited to a relatively small range of essential utilities.
- It is not difficult to envisage a similar schema being applied to a range of smart phones which could be distributed with an appropriate permuted operating system.

Published by Paul Cockshott

Political Economist and Computer Scientist View all posts by Paul Cockshott