Massimo Bernaschi
Istituto Applicazioni del Calcolo
National Research Council of Italy
massimo.bernaschi@cnr.it

Simplicity, Speed, Security, for Software Sustainability

The KISS principle

- The cheapest, fastest, and most reliable components are those that aren't there (G. Bell)
- Controlling complexity is the essence of computer programming (B. Kernighan)
- One of my most productive days was throwing away 1000 lines of code (K. Thompson)
- Simplicity is prerequisite for reliability (E.W. Dijkstra).

A simple problem

- Most Unix-like systems have a *words* file that is a list of English words (many times is /usr/share/dict/words)
- How do you find which is the word with more anagrams and count them?
  - example: *rescue* is an anagram of *secure*

A possible approach

1. Create a github repository
2. Set up *cmake* to generate native build tool files specific to your compiler and platform
3. Download and import tons of libraries *just in case*
4. Set up *docker* to deliver your solution in a container because *maybe one day* someone could need to update the operating system…

More seriously

- How much time is required if there are (about) 500000 words in the list?
- Time for

1. Finding a solution
2. Implementing the solution
3. Testing/debugging
4. Executing

A *very simple* solution

Sorting the characters in the words is the *key* to the solution!

```
for w in $(cat $1); do
        echo -n "$w "
        echo "$w" | fold -w1 | sort | tr -d '\n'
        echo " "
done | sort -k 2 | tee saveanagram.txt | uniq -f 1 -c | grep -v "1 " | sort -nr
```

If the file contained secure and rescue (one word per line) the loop produces

```
secure ceersu
rescue ceersu
```

Execution time? ~15 minutes on a good laptop

Is it a *safe* solution?

Homework: rewrite the program in a (different) scripting language

Can we do better?

```
while(fgets(line,sizeof(line),stdin)) {
        if(strchr(line,'\n')) {*strchr(line,'\n')=0;}
        fprintf(fpt,"%s\t",line);
        qsort(line,strlen(line),sizeof(char),ss);
        fprintf(fpt,"%s\n",line);
}
rewind(fpt);
i=0;
while(fgets(line,sizeof(line),fpt)) {
        aow[i]=strdup(line);
        if(aow[i]==NULL) {
            fprintf(stderr,"Could not get memory for aow[%d]\n",i); exit(1);
        }
        i++;
 }
 qsort(aow,n,sizeof(char *),sortstring);
```

Total number of lines: 75. Execution time?

Is it *safe*?

- C (and C++) are *notoriously* bad programming languages from the security viewpoint. Aren't they?
  - buffer overflows
  - strings manipulation
  - pointer arithmetics
  - *strange* integer conversion rules
- Simple solution: avoid C(++)! Can you?

Security vulnerabilities

- **CVE-2021-29921.** In Python before 3.9.5 the ipaddress library mishandles leading zero characters in the octets of an IP address string. This (in some situations) allows attackers to bypass access control that is based on IP addresses.
- **CVE-2021-3177.** Python 3.x through 3.9.1 has a buffer overflow in PyCArg_repr in _ctypes/callproc.c, which may lead to remote code execution in certain Python applications that accept floating-point numbers as untrusted input, as demonstrated by a 1e300 argument to c_double.from_param. This occurs because sprintf is used unsafely.
- **CVE-2016-7543.** Bash before 4.4 allows local users to execute arbitrary commands with root privileges via crafted SHELLOPTS and PS4 environment variables.
- **CVE-2019-5790.** An integer overflow leading to an incorrect capacity of a buffer in JavaScript in Google Chrome prior to 73.0.3683.75 allowed a remote attacker to execute arbitrary code inside a sandbox via a crafted HTML page.

Success stories (1)

- How can you mirror and synchronize a bunch of data?
- Remember, we look for efficient, secure, reliable solutions.

rsync

- *rsync* is an utility for efficiently transferring and synchronizing files between a computer and a storage drive and across networked computers (*Wikipedia)*
- Andrew Tridgell and Paul Mackerras wrote the original rsync (1996!). Current maintainer: Wayne Davison (one guy!)
- Two processes: a sender and a receiver
- At startup, an *rsync* client connects to a peer process (using ssh). Data in flight are encrypted.
- Three ways for being *efficient*

1. Determining which files to send
2. Determining which parts of a file have changed
3. Compressing data (optional)

rsync approach

- Determining which files to send
  - (by default) by checking the modification time and size of each file
  - resort to a slower but comprehensive check if invoked with --checksum
- Determining which parts of a file have changed (delta encoding)
  - The recipient splits its copy of file *foo* into chunks and computes two checksums for each chunk
  - The sender then sends the recipient those parts of the file *foo* that did not match, along with information on where to merge existing blocks into the recipient's version.
- Compressing data (optional)
  - When compressing is not a good option?
- *rsync* is written in C as a single threaded application
- Look at https://rsync.samba.org/security.html, very few, minor, problems in 20 years!

Success stories (2)

- How can you transfer data using URL syntax?
- That is you need to support DICT, FILE, FTP, FTPS, GOPHER, GOPHERS, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, MQTT, POP3, POP3S, RTMP, RTMPS, RTSP, SCP, SFTP, SMB, SMBS, SMTP, SMTPS, TELNET, TFTP!

cURL

- cURL is a command-line tool for getting or sending data including files using URL syntax
- cURL was first released in 1996. The original developer is Daniel Stenberg. More than 800 contributors!
- Mostly written in C (plus ~ 15% in Perl+M4)
- 7 vulnerabilities in 25 years. The last one 10 years ago
  - **CVE-2012-0036.** curl and libcurl 7.2x before 7.24.0 do not properly consider special characters during extraction of a pathname from a URL, which allows remote attackers to conduct data-injection attacks via a crafted URL, as demonstrated by a CRLF injection attack on the (1) IMAP, (2) POP3, or (3) SMTP protocol.
- On April 27 2022 (today!) released version 7.83.0 and **4** security advisories

Success stories (3)

- How can you manage data according to the relational model without installing a DBMS?
- D. Richard Hipp solved the problem more than 20 years by releasing ***SQLite*** a database engine written in C
- SQLite allows the program to be operated without installing a DBMS or requiring a database administrator
- Good in reading, poor in writing!
  - SQLite read operations can be multitasked, though writes can only be performed sequentially
- All historical vulnerabilities reported against SQLite require at least one of these preconditions:
  - The attacker can submit and run arbitrary SQL statements.
  - The attacker can submit a maliciously crafted database file to the application that the application will then open and query

But even good projects may fail… (1)

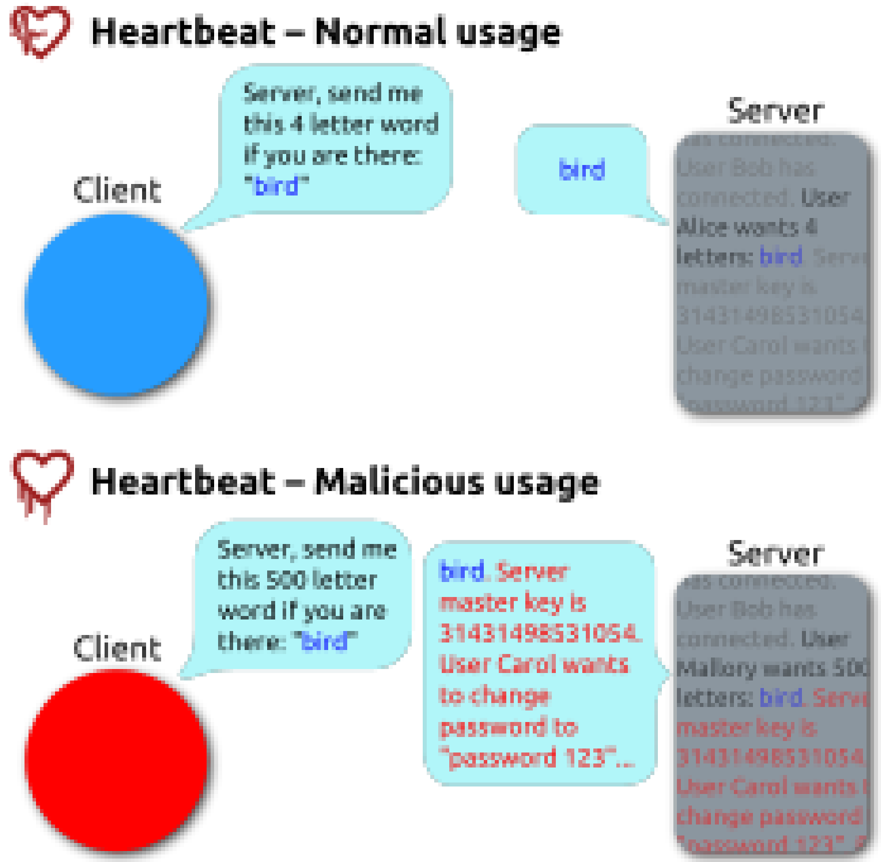
- Do you know the *Heartbleed* story?
- *Heartbleed* is a bug in the OpenSSL cryptography library (CVE-2014-0160). Found by Neel Mehta.
- The EFF and Bruce Schneier deemed the Heartbleed bug "catastrophic“
- How does it work?

1. A heartbeat request message consists of a payload, typically a text string, along with the payload's length as a 16-bit integer
2. The receiving computer then must send exactly the same payload back to the sender.

So what? It is a simple "echo"… what can be wrong with it?

The Heartbleed bug

- OpenSSL allocated a memory buffer for the message based on the length field in the requesting message


Heartbeat – Normal usage
Client
Server, send me this 4 letter word if you are there: "bird"
bird
Server
User Bob has connected. User Alice wants 4 letters: bird Server master key is 31431498531054. User Carol wants to change password to "password 123"
Heartbeat – Malicious usage
Client
Server, send me this 500 letter word if you are there: "bird"
bird. Server master key is 31431498531054. User Carol wants to change password to "password 123"...
Server
User Bob has connected. User Mallory wants 500 letters: bird. Server master key is 31431498531054. User Carol wants to change password to "password 123"

Picture from *Wikipedia*

The problem can be fixed by ignoring Heartbeat Request messages that ask for more data than their payload need!

Openssl has more than 350 CVE Records since 1999!

Other (not C(++)) success stories:

- Apache Cassandra (java), a free and open-source, distributed, wide-column store, NoSQL DBMS (7 CVE since 2015!)
- Angular (TypeScript) web application framework with a built-in policy for preventing cross-site scripting.

Despite the availability of many good tools and solutions we still have security and efficiency problems with software applications!

What do the *bad guys* develop in the meantime?

Ransomware!

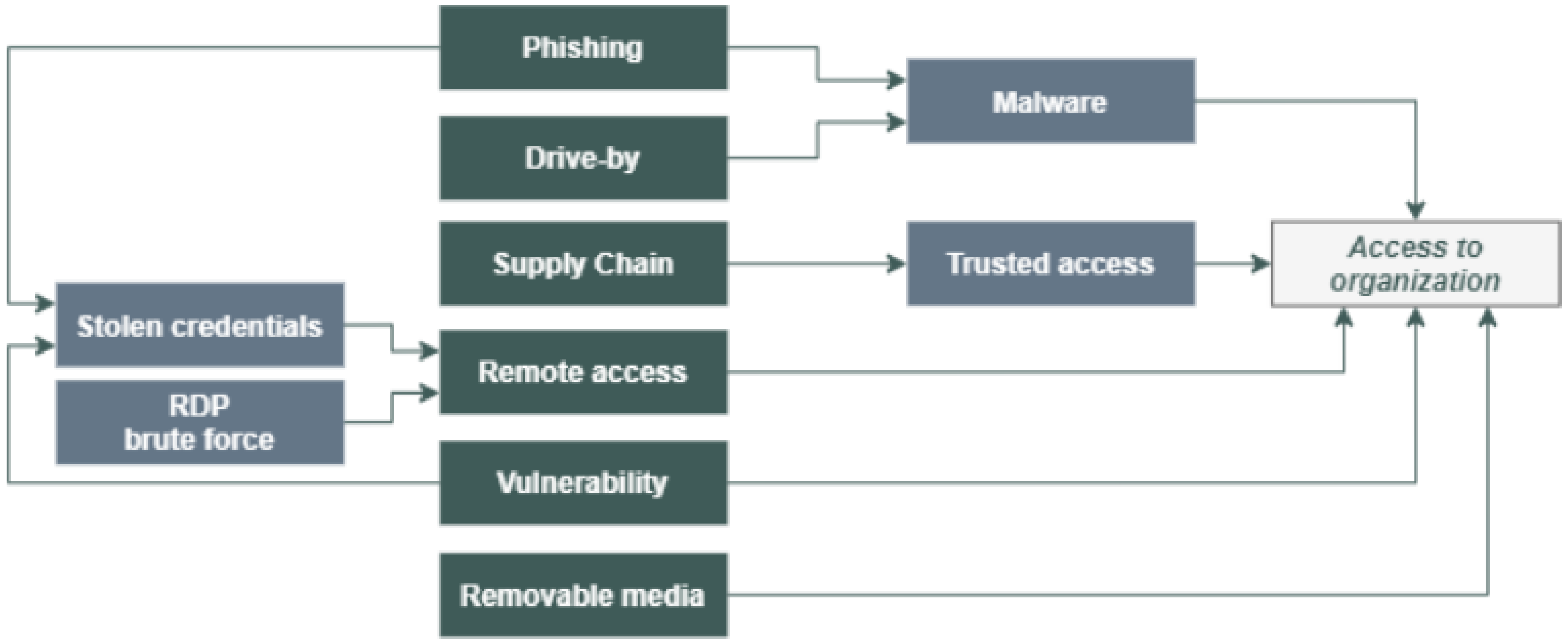
Why does ransomware work so well?

A ransomware is a malware that employs encryption to hold a victim's information at ransom.

A ransomware attack is carried in six stages

1. Access
2. Infection
3. Staging
4. Scan
5. Encrypt
6. Payday

Ransomware: access stage



From *THE ANATOMY OF TARGETED RANSOMWARE ATTACKS*
Denmark Center For Cyber Security

internal attacks are quite common!

What next?

The attackers start by downloading or configuring the tools they expect to use

- New malware or pen-testing tools
- Existing malware on the system
- Existing legitimate programs on the victim's computer.

Then they often scan the network to find ways to compromise other clients and servers (lateral movement)

- RDP and SMB (file sharing) are preferred targets.
- *Mimikatz* is a typical tool used to extract password in clear from a computer's memory

Then they establishing other entry points into the victim's network to safeguard their presence (*backdoors*)

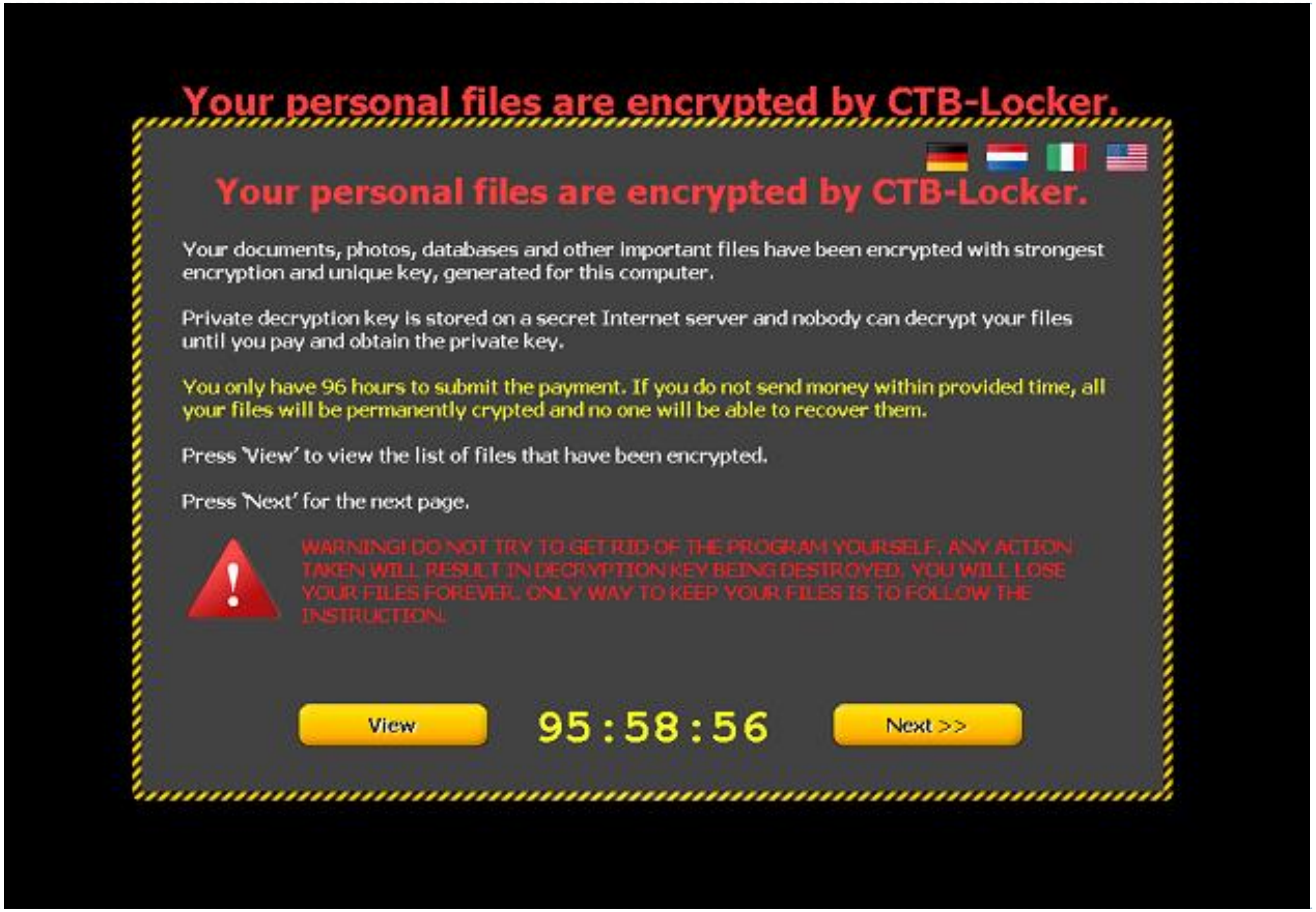
The best is yet to come… (1)

- If attackers gain administrator access to a Domain Controller they can deploy ransomware to the entire domain in one shot!
- *BloodHound* is a tool capable of quickly mapping an organization's hierarchy of privileged accounts
- Once the attackers have the domain administrator rights, they look for (and destroy) backups!
  - Local backup (shadow copies) may be easily deleted with VSSADMIN.EXE or WMIC.EXE standard programs
  - Central backup systems maybe more or less *safe* depending on the configuration but in general they have been designed and deployed with other requirements and goals.

The best is yet to come… (2)

- More recently, there is a new threat associated to ransomware attacks, that is the *exfiltration of sensitive data*
  - Attackers threaten to release sensitive data on public Internet sites if the victims fail to pay ransom
  - *Sidoh* is a espionage software which only gathers information and does not hold it to ransom
- Usually, before starting the encryption, the attackers turn off security mechanisms (antivirus, etc…) and, at times, services that could block the encryption (e.g., DBMS).
- Encryption does not take a long time and the final result is…

A very clear message…


Your personal files are encrypted by CTB-Locker.

Your personal files are encrypted by CTB-Locker.

Your documents, photos, databases and other important files have been encrypted with strongest encryption and unique key, generated for this computer.

Private decryption key is stored on a secret Internet server and nobody can decrypt your files until you pay and obtain the private key.

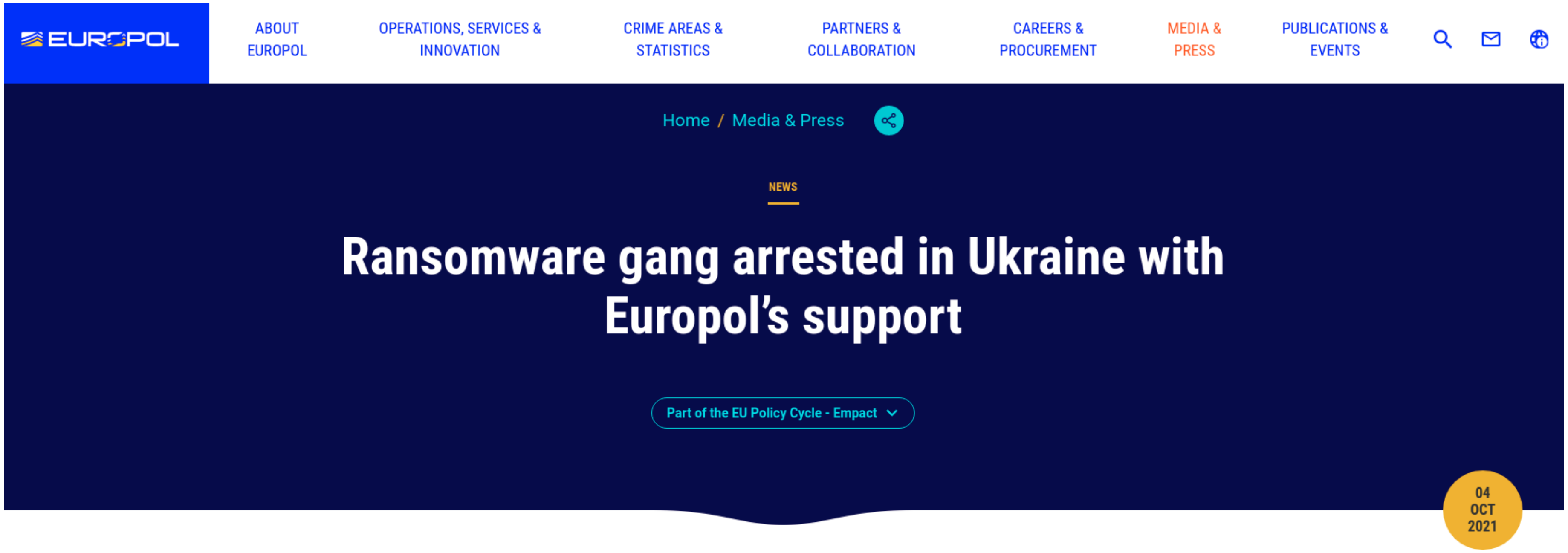
You only have 96 hours to submit the payment. If you do not send money within provided time, all your files will be permanently crypted and no one will be able to recover them.

Press 'View' to view the list of files that have been encrypted.

Press 'Next' for the next page.

WARNING! DO NOT TRY TO GET RID OF THE PROGRAM YOURSELF. ANY ACTION TAKEN WILL RESULT IN DECRYPTION KEY BEING DESTROYED. YOU WILL LOSE YOUR FILES FOREVER. ONLY WAY TO KEEP YOUR FILES IS TO FOLLOW THE INSTRUCTION.

But also…


ABOUT EUROPOL
OPERATIONS, SERVICES & INNOVATION
CRIME AREAS & STATISTICS
PARTNERS & COLLABORATION
CAREERS & PROCUREMENT
MEDIA & PRESS
PUBLICATIONS & EVENTS

Home / Media & Press

NEWS

Ransomware gang arrested in Ukraine with Europol's support

Part of the EU Policy Cycle - Empact

04 OCT 2021

This operation has been made possible thanks to a technical solution developed by a Europol's senior specialist who received his PhD in computer science at Sapienza.

Is decryption possible?

- Encryption used by ransomware is, in general, very strong
- Very few chances of decrypting files. However, it is worth to take a look at https://nomoreransom.org (by Europol)
- It is apparent that systems restoration is preferable to the payment of any ransom. However…
  - Risk of copying the backdoors during the restoration process
  - Risk of deleting evidence of the intrusion.
- Do we have any hope?

Pain points in the current situation

- Very (too much?) complex operating environments
- (so far) Main targets are Windows environments
  - The problem is **not** the operating system!
- There are tons of recommendations and one should follow all of them, but this is not enough
- Lack of simple, specific, sustainable solutions
- What is the *key* to the solution?

First of all: KYD (Know Your Data)

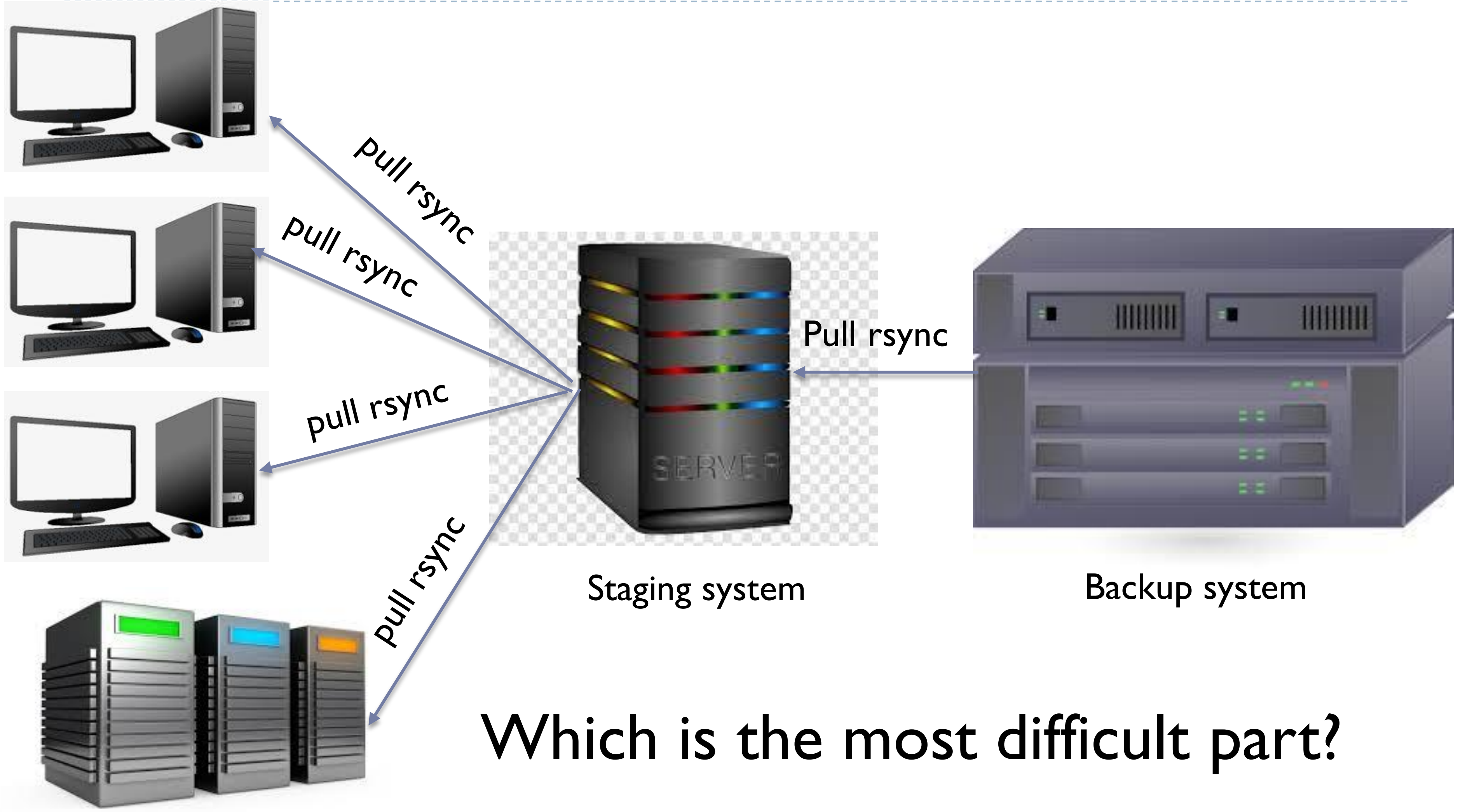
- It is unlikely that in the short/medium term we will be able to prevent systems from begin compromised
- How do we minimize the damage if a ransomware infects our systems?

1. Different policies for software and data
2. Priority to data integrity
3. (Optional) define which data you want to protect
4. What should you avoid?
5. How do you keep it simple?

Second move: Invert the logic

1. Systems should not send data to the backup system
2. A staging system (with no other Internet exposed service running) should pull selected data from the systems
3. Data should be *validated* on the staging system (e.g., no encrypted data unless they are also signed with a specific key)
   - Data that do not pass the validation stage might trigger an alarm
4. The (real) backup system store validated data only
   - Many possible variants: incremental copies, periodic full copies, *etc*…

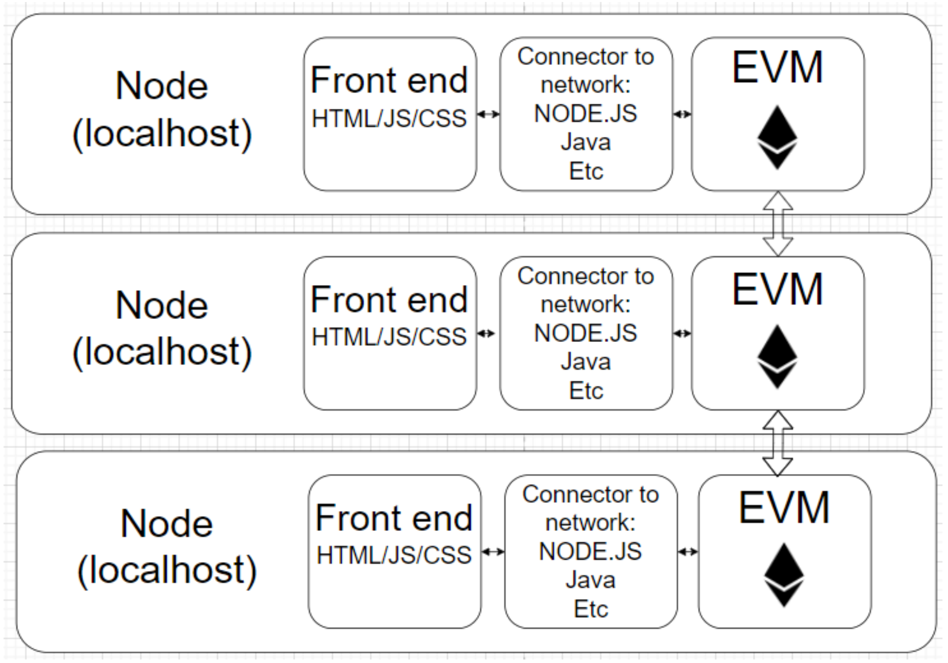
Pulling data to safely backup them



Which is the most difficult part?

Why all of this stuff is relevant

- A software based economy is made possible by smart contracts (and cryptocurrencies)
- We might shift from an enterprise based to a blockchain based model



(Picture from https://dzone.com/articles/comparing-blockchain-and-classical-enterprise-arch)

Simplicity, Security, Speed

- The cost of fixing *solutions* which do not consider efficiency as a requisite is daunting
  - Think to Ethereum switch from PoW to PoS!
- Don’t be fooled by Sir Tony Hoare’s quote

premature optimization is the root of all evil

(read *The fallacy of premature optimization* by Randall Hyde)

- Complex solutions increase the chance of security issues
  - A rough estimate is that the number and the severity of security issues increase as the square of the complexity

Good code is short, simple, and symmetrical – the challenge is figuring out how to get there (Sean Parent).