Tales from cloud nine

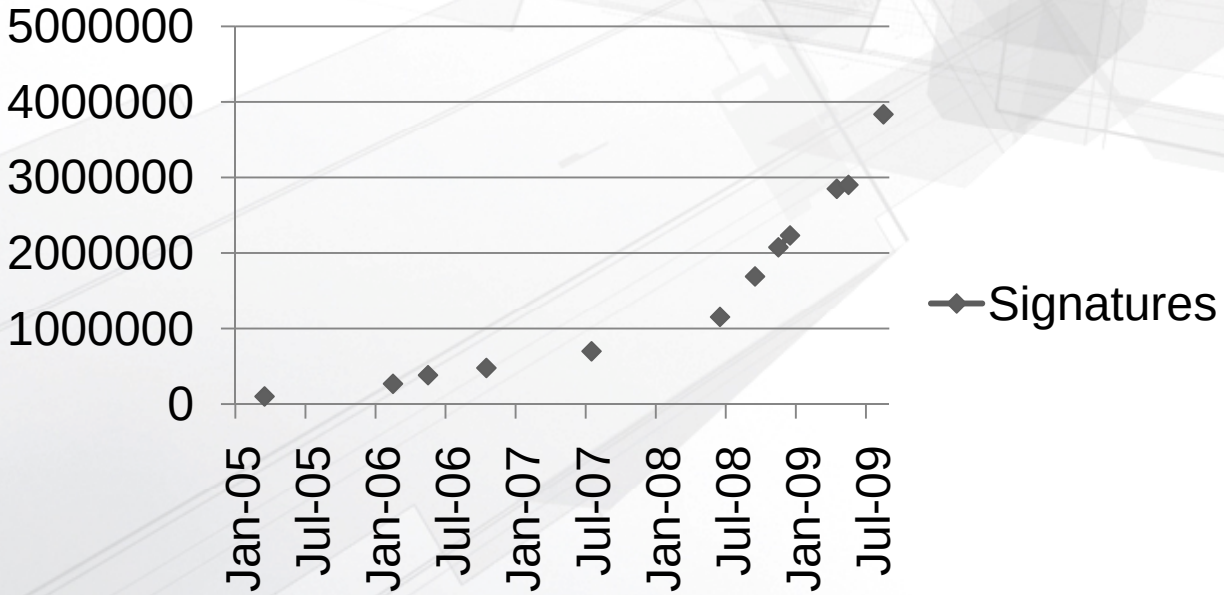
Mihai Chiriac,
BitDefender

Talk outline

- Motivation
- Technical challenges
- Implementation results
- Future ideas
- Conclusions

Reasons

- Malware numbers have grown at exponential rates

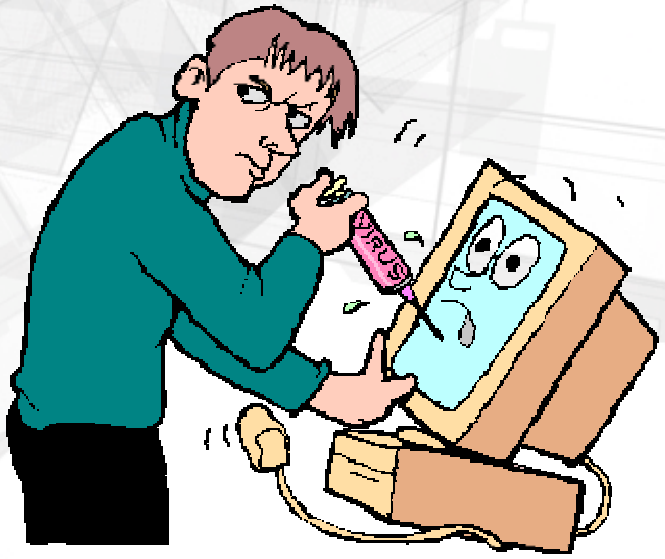


Reasons

- Bloated local virus signature databases…
  - Large memory usage (60 – 200 MB)
  - Slower loading times
- Harder to deliver virus updates
  - Hourly updates…
  - Bandwidth cost ☹
- Not a big problem for now, but we expect 10M signatures in 2010 !

Virus writer's profile, 1989-1999

- Primary reasons
  - Underground recognition
  - Fame (newspapers etc)
  - Boredom
  - "Playing God"
  - "Getting even"
  - …
- Typically independent



Virus writer's profile, 1999-2009

- Malware writing became an industry
  - Source control software (CVS etc)
  - Release cycles
  - Updates
- Purely financially-driven
  - Bot nets sold for spam, DoS
  - Identity & data theft
  - Ransom

Idea

- Keep as much as possible of the malware DB on the AV company's servers
- Provide same performance
  - Same detection/FP rates
  - Similar speed

2M vs. 7M?

- Signature numbers vary between 2 and 7 million, depending on vendor
  - Detection method
  - Heuristics?
  - Signatures or generic routines?
- Signatures are important
  - Exact identification: support, prioritization, remediation

Malware detection types

Detection types – non executable

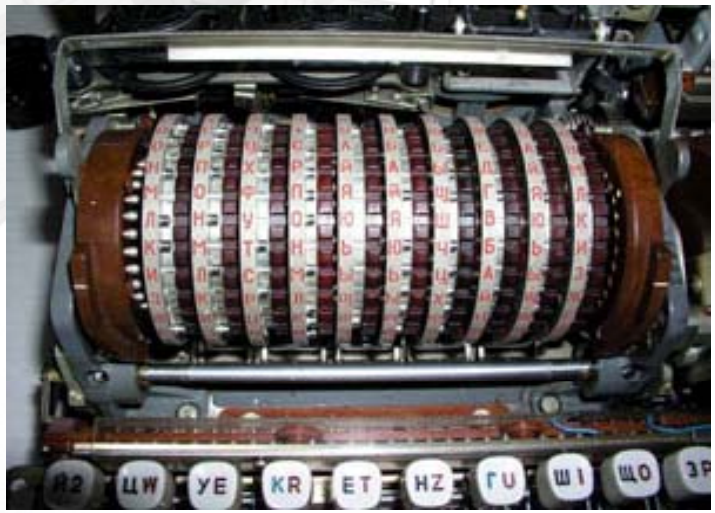
- Complex file format parsers, interpreters
- Less than 3% of signatures
- Client-server scanning is inefficient
- Privacy issues when uploading chunks of files for scanning
- *It's best to keep detection engines and signatures locally*.

Detection types - executable

- String search
  - Mostly based on Aho-Corasick
  - Boyer-Moore, KMP, also used
  - Only used in extreme cases (corrupted or too short viruses, weird file formats, etc)
  - 0.1% of virus DB size
  - *Should run on the client*

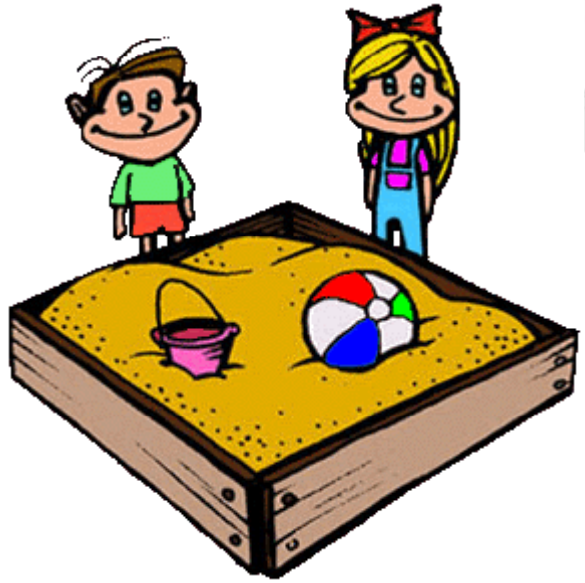
Detection types - executable

- X-ray (cryptanalysis)
  - Plaintext attack against encrypted data
  - Some X-ray algorithms are CPU intensive
  - Needs the buffer…
  - …and the plaintext data ☺
  - 1.6% of signatures…
  - *Should run client-side*



Detection types - executable

- Sandbox & generic detection routines
  - Need file chunks
  - 50+% of all new malware is detected generically
  - CPU intensive task
  - *Should run on the client*

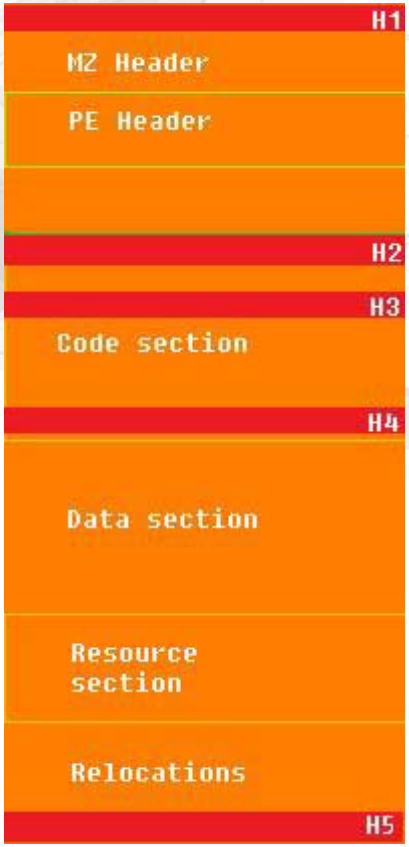


Detection types - executable

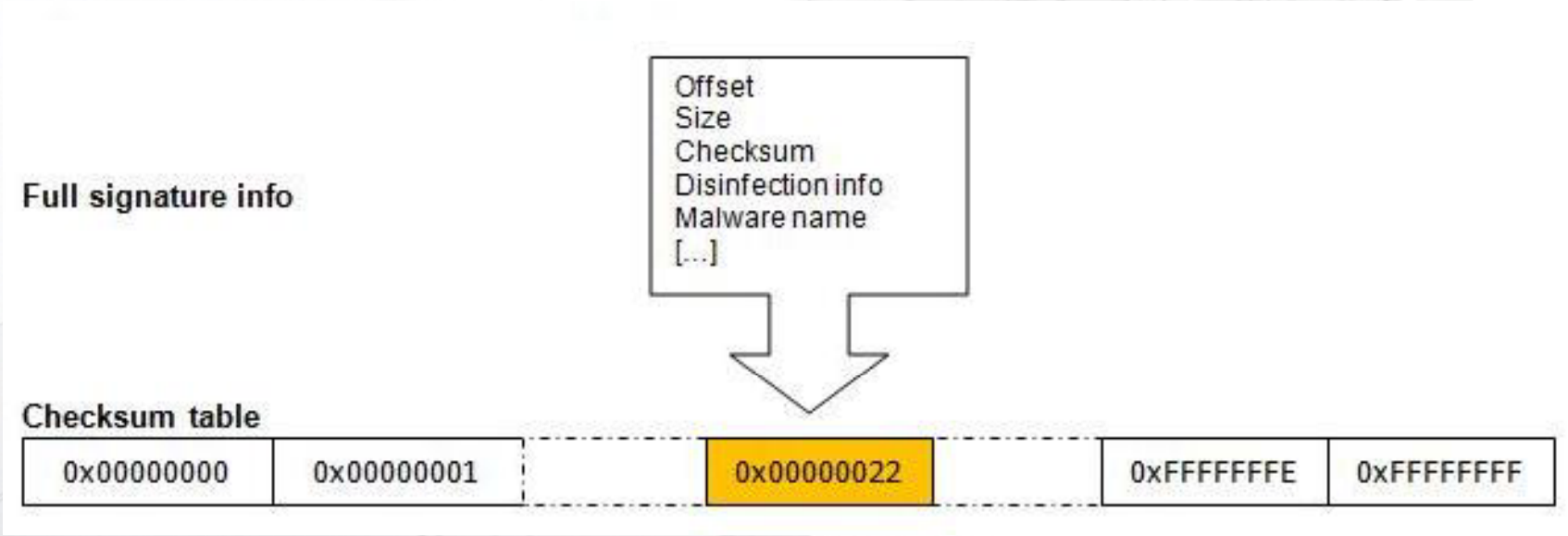
- Static malware
  - Does not change over time
  - Detected with checksums at fixed offsets, fixed sizes
  - 90.9% of the BitDefender malware DB size!
  - Easy to identify a file via its crypto hash
  - Easy to port to a client-server architecture!!

Static malware detection - 1

- Parse the file, extract N zones
- Perform the N checksums
- Search for the checksums
  - "not found" – clean
  - "list of additional blocks"
- Perform additional checksums
  - Verify the results



Static malware detection - 2



Static malware detection - 3

- We can keep the static malware DB on our servers
- Instant updates for static malware
- Same detection rate, 10% DB size!
- Malware DB becomes a dynamic entity
  - Easy to spot “busy” checksums
  - Easy to change/optimize signatures

Static malware detection - 4

- Checksum queries can be expensive
- Local copy of the checksum table?
  - Only a fraction of the full DB (memory, bandwidth for updates, etc)
  - Query the server only when we have a checksum “hit”
- …Can be a solution in case malware numbers continue to grow at this rate…

Problems

- Disconnected operation
- Client-server approach needs specialized content-delivery systems
  - Standard methods won’t work
- Privacy
  - Only checksums, but they’re enough to identify files ☹

Conclusion

- Same detection rate, ~20% DB size (with local hash table)
- *…but if we only scan a few files we don't need the local hash table!!* ☺
- Less than 20% bandwidth for DB updates
- Several issues need to be addressed
- It's a possible solution for the future

Implementation

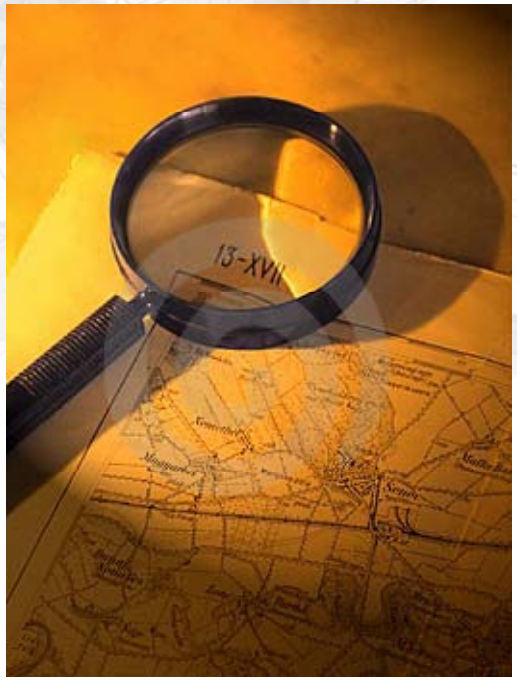
- We’re scanning “a normal system”, and not a “virus collection”
- Scan only the system’s “sensitive areas”
- The vast majority of malware is detected!

Sensitive areas - 1

- All executable modules from all processes (on-disk and in-memory)
- All kernel modules
- All system services (regardless of their execution state)
- All system entry points
- All BHOs, browser add-ons etc (even if the browser is not loaded)

Sensitive areas - 2

- All sensitive registry keys & associations
- All modules registered as LSPs, Winlogon notify DLLs…
- All files present in sensitive folders
- …



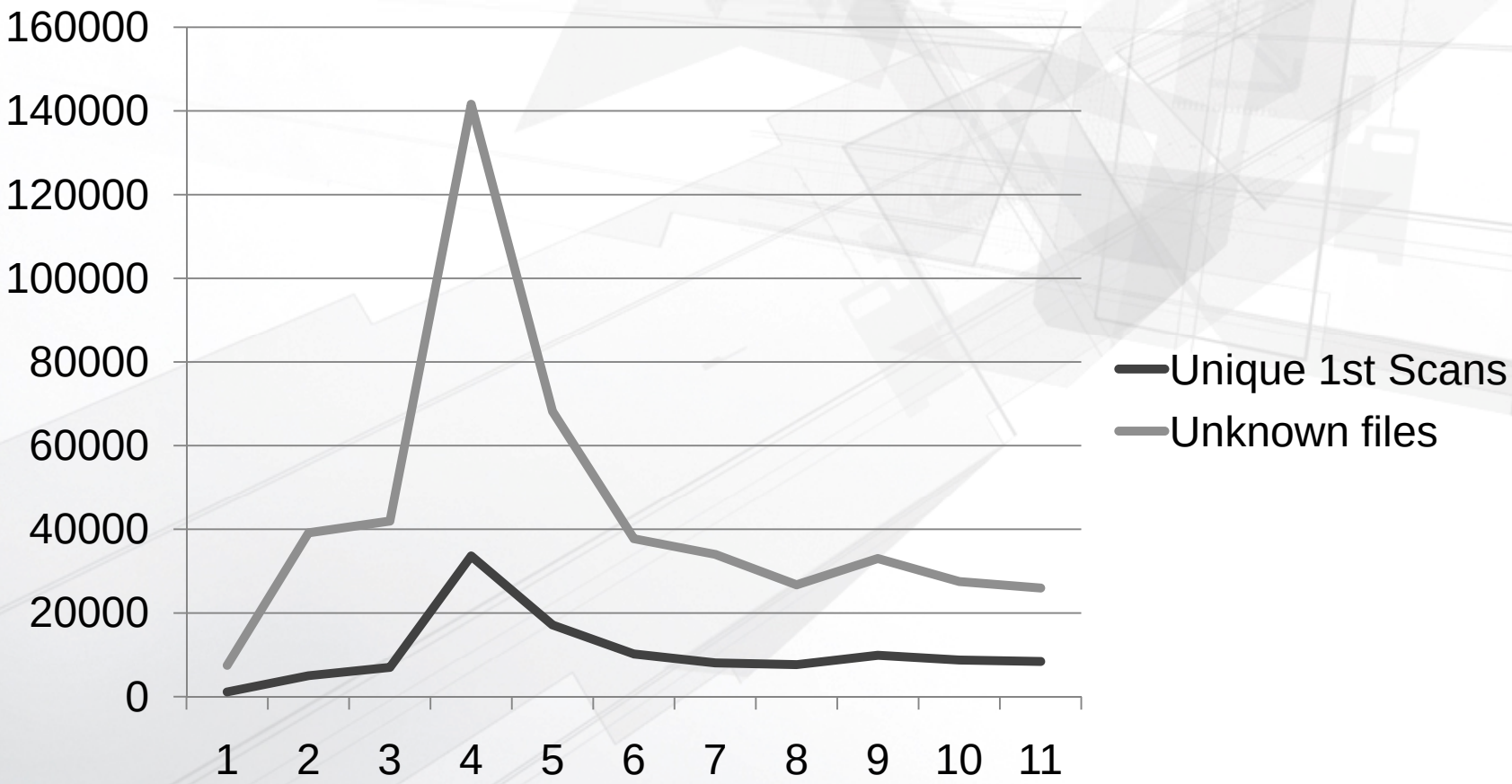
First statistics

- An average of 685 “sensitive” files to scan
- 675 for Windows® XP machines
- 755 for Windows® Vista machines
- The majority are known-clean files…
  - identify them (on disk & in memory) and exclude them…
  - continuously scan these files server-side
- What about the files that are unknown?

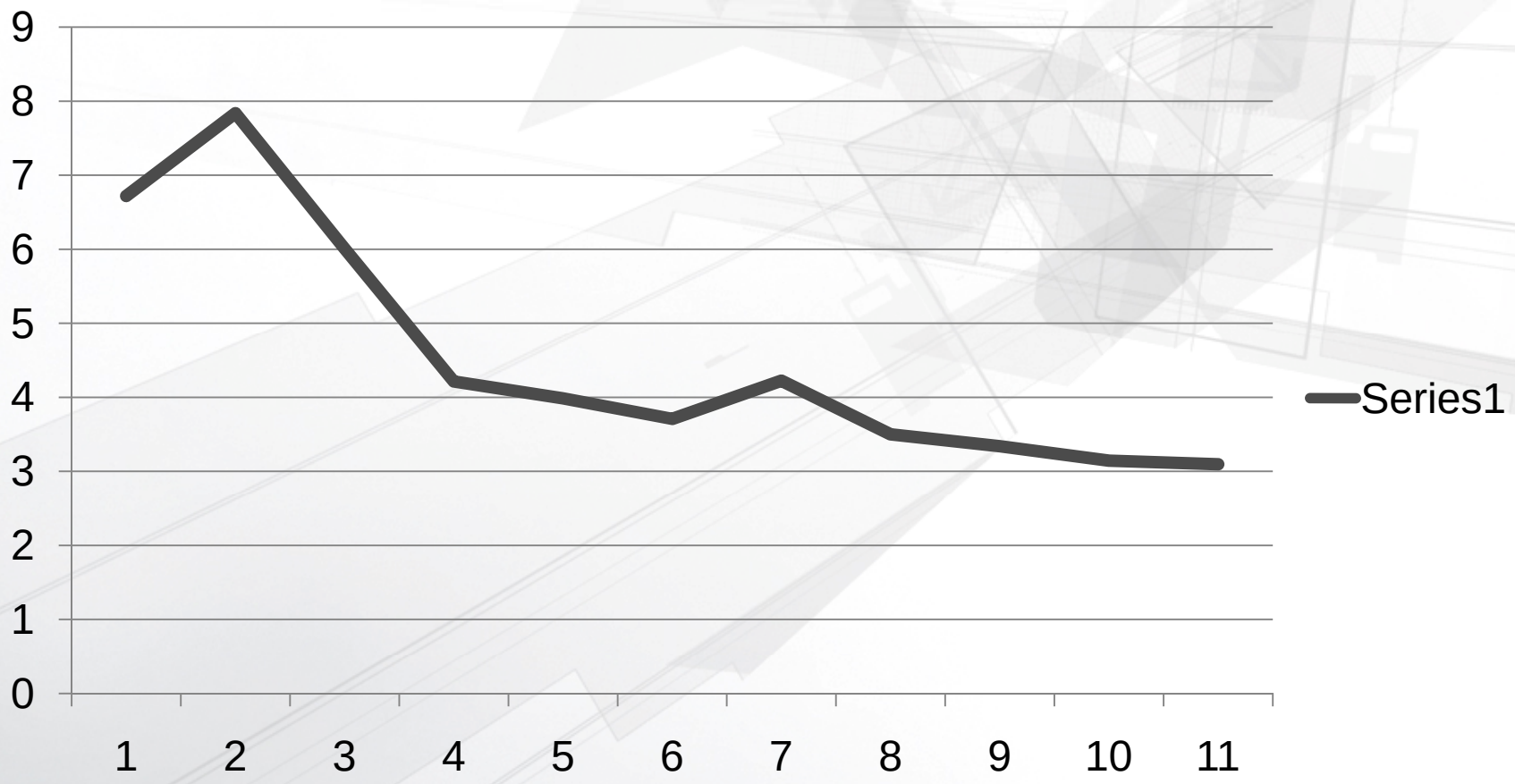
“Unknown files” - 1

- Static malware – client/server scanning
- Non-static malware:
  - 1. “Force” static detection
  - 2. Download non-static detection engines + signatures
  - 3. Upload all unknown files (or file chunks)
  - 4. Upload only “suspicious” files

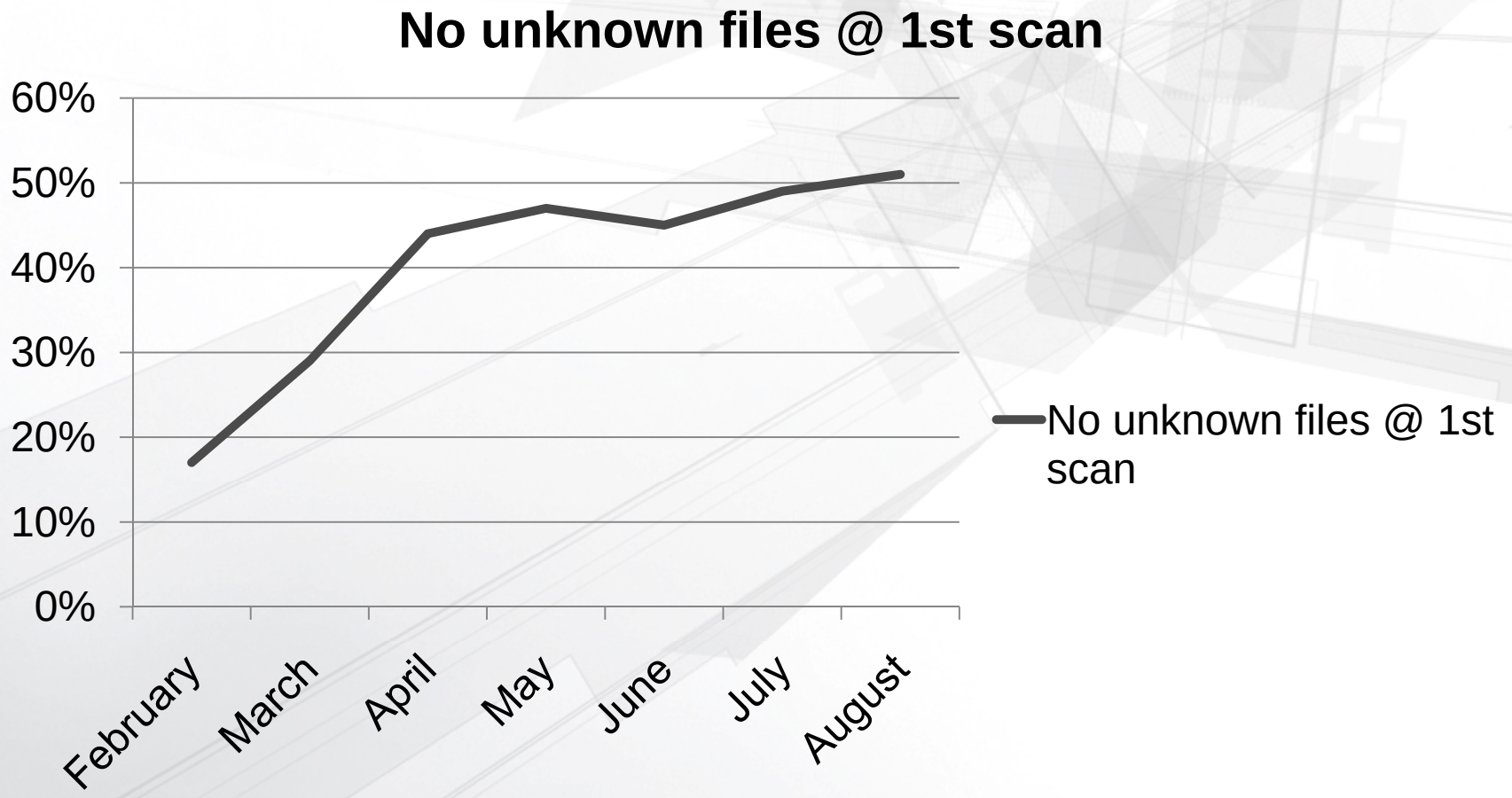
“Unknown files” - 2



“Unknown files” - 3



“Unknown files” - 4



“Unknown files” - 5

- Average file size – 220 KB
- The average user can upload ~660 KB or download ~5 MB
- Upload:
  - Only once, other users will benefit
  - Less bandwidth (although upload may be slower than download on some connections)
  - Some privacy issues

“Unknown files” - 6

- Upload the entire file
  - We can continuously rescan it server-side
  - We can use the file in our collections (both white and black lists) and tests
- Upload only file chunks
  - We only need an average of 18 KB from each file!
  - New detection routines will possibly need new file content ☹

Results - 1

- Average 1st scan took 59s
- Average 2nd scan took 29s
- Quick diagnostic tool?
- NAC device?
- Etc…

Results - 2

- ~20% of users were infected
- Non-static malware
  - 10% of all detected malware!
  - Samples seen *only once*!
  - “Forced” static detection won’t work

Forensics analysis

- Thousands of new computers per day
- At least 1 in 5 is infected!
- We can extract info from the possibly compromised PCs!

In theory…

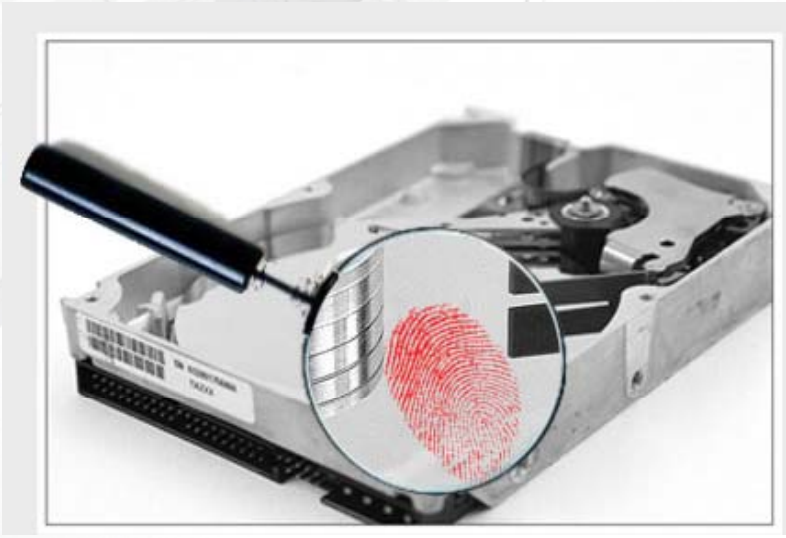
- We’ve already lost the game
- The machine is compromised
- Malware is already running!
- We only “see” what it allows us to see!

In practice…

- No such thing as “perfect stealth”
- Non-perfect stealth is a dead giveaway!!
- Many useful informations
- Simple process (I.M.D.)
  - Collect intelligence
  - Send it to the cloud
  - Parse the data using rules
  - Use the result for prioritization

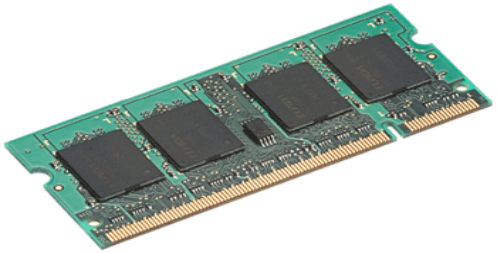
File informations

- BitDefender “SimFS” raw file system parser
  - Allows access to locked files
  - Detects hidden files
- File content analyzer
  - PE structure, imports
  - Entropy
  - Digital signature



Module informations

- Already running
  - Probably already decrypted!
- We can search for “clues”
  - Exploits and shellcode
  - Embedded PE files (device drivers?)
  - Strings used by interesting protocols, etc
- On-disk vs. in-memory image
  - Packed module?

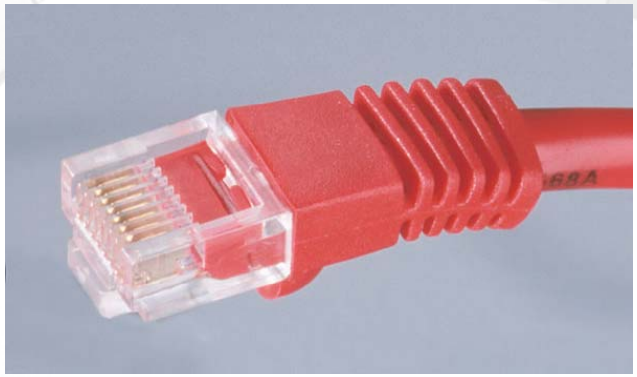


System informations - 1

- Hidden process detection
  - A “detected” hidden process is a dead giveaway!
- Hidden module detection
  - Legally loaded module, but removed from PEB In*Order lists
  - Manually mapped module (orphan code)
  - Very strong flags!

System informations - 2

- Open ports
  - Process connected to server, port?
  - Process listening on a specific port?
- API Hooking
  - Target of an API hook?
- Keystroke logging?
- (Auto)execution method
  - Parent process?

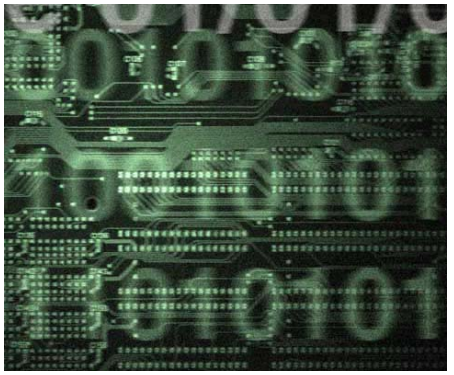


System informations - 3

- “Sensitive” registry keys
  - AppInit_DLLs? Winsock LSP?
  - Hidden? Even better ☺
- “Sensitive” file system areas
  - StartUp? Autorun.inf? Streams?
- “Sensitive” processes
  - Hidden/packed/unsigned module loaded inside a system process?

Data processing

- “IMD Packets”
- A packet contains file, module, “system” informations
- Processed by a series of “IMD Rules”
  - Manually added, for now
  - To be automated…
- Helps in sample prioritization
  - …and maybe more?



Automatic blacklisting - 1

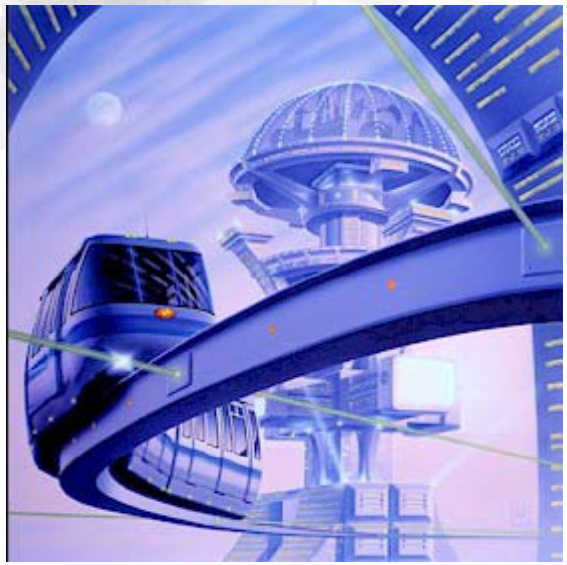
- Any single rule is not enough
- File info
  - Any file may be locked
  - ~20% of executables are packed…
- Memory info
  - Shellcode in memory image of any vulnerability scanner, AV, etc…
- System info

Automatic blacklisting - 2

- We use a combination of rules
  - Distribution, spreading info
  - Geographic data
- Sets of “IMD Flags” can be seen as a pseudo-signature
  - Group malware into families
  - Blacklist sets of flags

Future ideas

- Add more flags
  - From AVC (behaviour-based analyzer)
  - From B-HAVE (sandbox)
- Automate detection rules
  - Neural network?
- New idea: file “similarity”
  - Still to be analyzed…



Conclusions

- Right now, local scanning still works best for the general case
- A “best of both worlds” solution can be employed in the future
- Specific instances where cloud-scanning is more efficient
- Cloud intelligence can be extremely helpful ☺

Q&A

mchiriac@bitdefender.com

bitdefender
secure your every bit