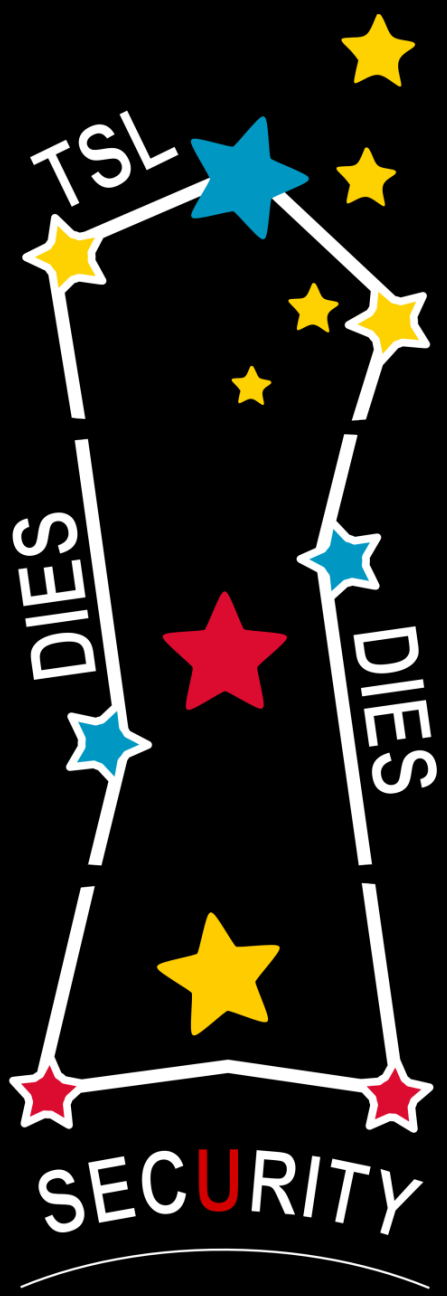
UNIVERSITY OF TWENTE.



GOODWARE DRUGS FOR MALWARE: ON-THE-FLY MALWARE ANALYSIS AND CONTAINMENT

DAMIANO BOLZONI

CHRISTIAAN SCHADE

TWENTE SECURITY LAB

UNIVERSITY OF TWENTE

THE NETHERLANDS

AGENDA

- Something about malware
- Malware internals
- Current analysis tools
- Our idea: on-the-fly malware analysis
- The Avatar architecture
- DEMO
- Conclusion

SOMETHING ABOUT MALWARE

- In the last half-decade malware has evolved into a business for cyber criminals
  - it is one of the most pressing security problems on the Internet
- Symantec and its friends show impressive statistics of growing rate, mainly due to:
  - polymorphism
  - packers

MALWARE INTERNALS

- Malware writers first shipped “monoholitic” executables
  - difficult to “adapt” to any OS configuration
  - easier for an AV to spot
- ~30% of current malware download additional components once running
  - a “spore” is responsible for “planting” the malware
  - downloaded components are used to collect username/ password, infect other EXEs, etc.
  - BOTnets are a classical example

CURRENT ANALYSIS/DEFENSIVE TOOLS

- Dynamic malware analysis (DMA)
  - malware samples are executed in a **sandbox** → every action performed is logged
  - some tools support **clustering** → detects a new sample from a known family
  - Anubis, CWSandbox, Malheur, Malnet
- Signature- and "model"-based AVs
  - DMA analysis reports are used to update signatures/models

PROBLEMS WITH DMA – 1

- Malware writers know about DMA tools, and implement several countermeasures to avoid/slow down analysis
  - runs only when users are actually logged in
  - waits for a certain time frame before activating
  - checks for virtualization
  - checks for known registry keys
  - check for known IPs

- A DMA tool lacks the execution context

PROBLEMS WITH DMA – 2

- DMA tools perform **only post-mortem** analysis → users submit their sample(s) and get a report back
  - limited support to monitor an internal network and protect end hosts
  - if you submit a sample, you already suspect it is malware…and your AV likely did not detect it (otherwise…why submit it for further analysis?)

- No real-time protection, as analysis requires special instrumentation

GOALS

- **G1**: Can we use dynamic analysis tools to perform *on-the-fly* malware analysis and *containment at the end host* without having to deploy any software component *before hand* ?

- **G2**: Can we create a NOC for malware ?

We call this architecture *Avatar*!

THE IDEA

- As malware downloads additional components, it requires some external “content providers” (usually early compromised web servers)

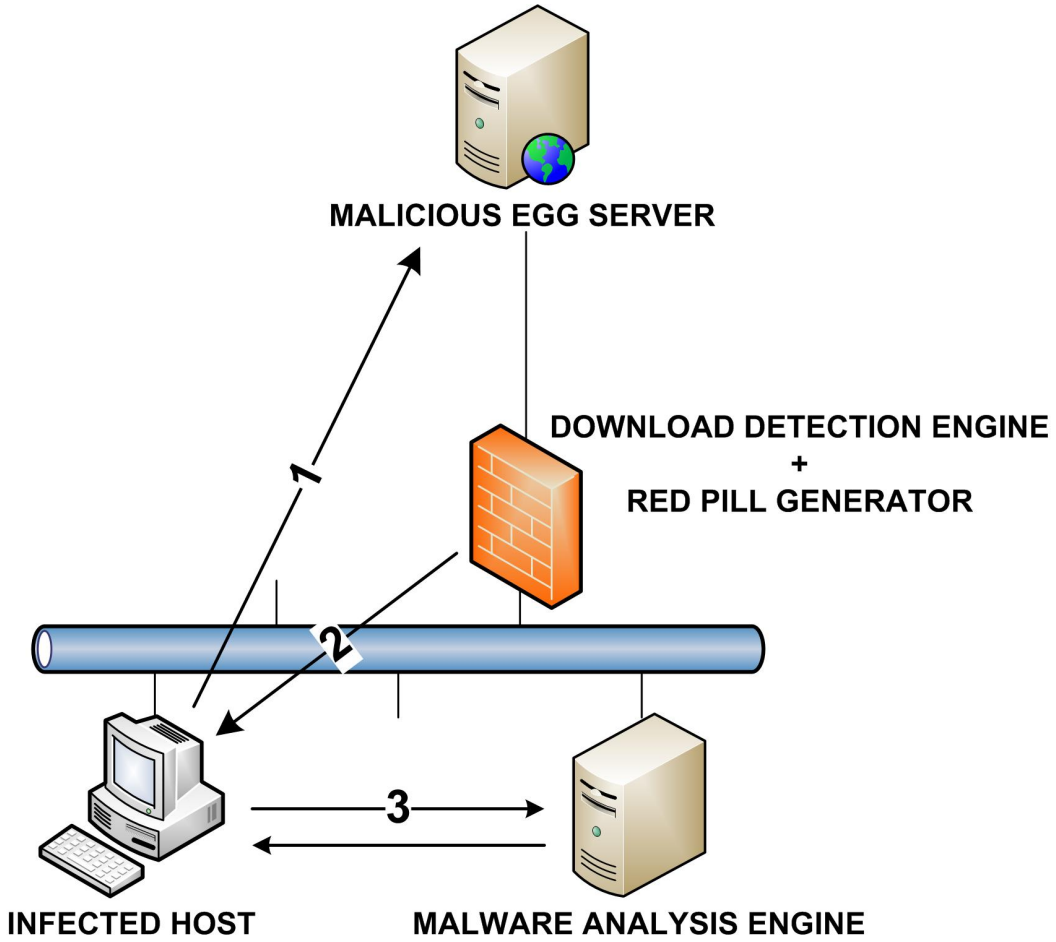
- Because such providers are not always available, malware runs several download attempts

- If we can detect one of these attempts, we can feed the malware with a crafted executable (we call it “red pill”) that:
  - will run some real-time analysis at the end host → on-the-fly malware analysis
  - can be instructed to terminate its parent process → effective containment

AVATAR – MAIN COMPONENTS

- We need at least 3 logical components
  - download detection engine (DDE) → detects failed download attempts
  - red pill generator (RPG) → packs the red pill and sends it back to the target
  - malware analysis engine (MAE) → receives information from the red pill, once this is executed

AVATAR – GENERAL ARCHITECTURE



IMPLEMENTATION – 1

- For practical reasons, we have implemented the DDE and the RPG into a single Linux box

- An iptables rule transparently re-route outgoing HTTP traffic to an Apache web server, working in proxy transparent mode. We developed an Apache module that:
  - uses an algorithm based on TWR to detect “too many” failed attempts
  - checks the requested filename
  - checks magic numbers in case a file is successfully fetched after several attempts
  - packs and sends the red pill when # attempts > threshold

IMPLEMENTATION – 2A

- When the red pill is executed on the target machine, it attempts to get control over its parent process by trying several access :
  1. PROCESS_ALL_ACCESS → full control
  2. TERMINATE_PROCESS | QUERY_INFO | READ
  3. QUERY_INFO | READ
  4. TERMINATE_PROCESS → least access rights

- Depending on the access level, and the OS version (latest 64 bit Windows versions allow fewer interactions), the red pill can:
  - ❑ freeze the process
  - ❑ terminate the process

IMPLEMENTATION – 2B

- The red pill collects then several information about the parent process:
  - path to the exe
  - any module that was loaded (full paths to the modules)
  - window (if any is attached) information: handle, size, caption text
  - executable size

- Collected information are sent back (encrypted) to the MAE, which determines whether to stop the red pill or perform deeper analysis
  - the red pill can send back to the MAE the original parent executable

IMPLEMENTATION – 3

- The MAE performs a thorough analysis
  - real box, no virtualization/emulation → avoid malware countermeasures against analysis tools (our goal is not to analyze as many samples as possible)
  - kernel driver → difficult to detect
  - can also interact with other dynamic analysis engines (Malheur)

WORKING MODES – TRANSPARENT MODE

1. The DDE notifies the RPG about the failed attempts

2. ONLY if a file is successfully downloaded, then the red pill is shipped

3. Provided the requested file is an executable, it is “glued” to the red pill so that it is executed once the red pill has finished the analysis

4. The red pill does not freeze or terminate its parent process, runs the preliminary analysis and, based on it, could send back to the MAE a copy of the parent executable

WORKING MODES – SEMI-TRANSPARENT MODE

- The DDE notifies the RPG about the failed attempts
- The RPG waits for the requested file to be pulled down, checks whether it is an executable, and ships the red pill with the original file
- The red pill freezes its parent process, runs the preliminary analysis and, based on it, could send back to the MAE a copy of the parent executable
- When the MAE sets a verdict about the parent process, the red pill releases or terminates it

WORKING MODES – NON-TRANSPARENT MODE

- The DDE notifies the RPG about the failed attempts
- Provided the requested filename points to an executable, the RPG sends back a red pill right away
- The red pill runs the usual checks, possibly sends the parent executable, and freezes the parent process
- When the MAE sets a verdict about the parent process, the red pill releases or terminates it

LIMITATIONS – 1

THERE ARE SOME LIMITATIONS TO OUR APPROACH

- Because we use some statistics-based heuristics to detect failed download attempts, malware could initiate connections at a very low rate → this would slow down the infection though

- Malware could apply some verification/encryption mechanisms to the downloaded components → this would make updates more difficult (keys/hashes would have to be known in advance) or could be broken as the malware become known

LIMITATIONS – 2

- Malware writers could use steganography to hide executables into other file formats (e.g., JPEG) → we could add some plug-ins to verify that format matches content

- Malware could leverage the CreateThread function to execute its code into another process → this could mislead the information collected by the red pill about the parent executable

TESTS

- The Avatar approach has been tested against real-life malware samples
  - CWSandbox data set, available at Malheur's web site
  - Everyday malware we all receive in our mailbox ☺

- Dataset A
  - ~10 malware families, huge collection (almost) publicly available from the authors of Malheur (2009) → 75 samples

- Dataset B
  - Everyday malware we received in our mailboxes during a week time (2010) → 30 samples

TEST RESULTS – DATASET A

Malware family	# of samples	# of samples marked as anomalous by DDE (red pill was shipped)	# samples that actually executed the red pill
Agent	9	9	9
Adload	8	6	6
Banload	3	2	2
Chifrax	2	2	2
FraudLoad	8	5	4
Genome	4	4	4
Geral	9	8	8
Killav	6	5	0*
Krap	6	4	4
NothingFound	10	10	3
Xorer	7	6	4

TEST RESULTS – DATASET B

	# of samples
Malware samples correctly identified by DDE	28/30
Malware samples that executed the red pill	27/30
Malware samples correctly identified by heuristics	13/30
Malware samples erroneously identified as goodware by heuristics	2/30
Malware samples sent to MAE for analysis	12/30
Non-malware samples erroneously identified by DDE	10/30
Non-malware samples correctly identified by heuristics	6/30
Non-malware samples erroneously identified as malware by heuristics	2/30
Non-malware samples sent to MAE for analysis	2/30

DISCUSSION

- No “sanity| check is basically run on the downloaded file
  - malware executes it right away
- The heuristics are usually enough to determine whether a running program is malware
  - ~50% of malware detected by the heuristics
- Some samples did not execute the red pill
  - they act as bogus “download service”, leaving the last step of actually launching the malware up to the user

DEMO

- Show time!

CONCLUSION

- Avatar raises the bar of malware analysis
  - no software is required to run at the end host
  - Avatar delivers on-the-fly any component needed for analysis
  - heavy computations are off-loaded
  - we can stop a malicious process as soon as it is detected (to some extent, depending on the OS)

- We know it can be avoided, but this will also make it more difficult for malware writers
  - no countermeasure has been observed so far in our tests

QUESTIONS

?