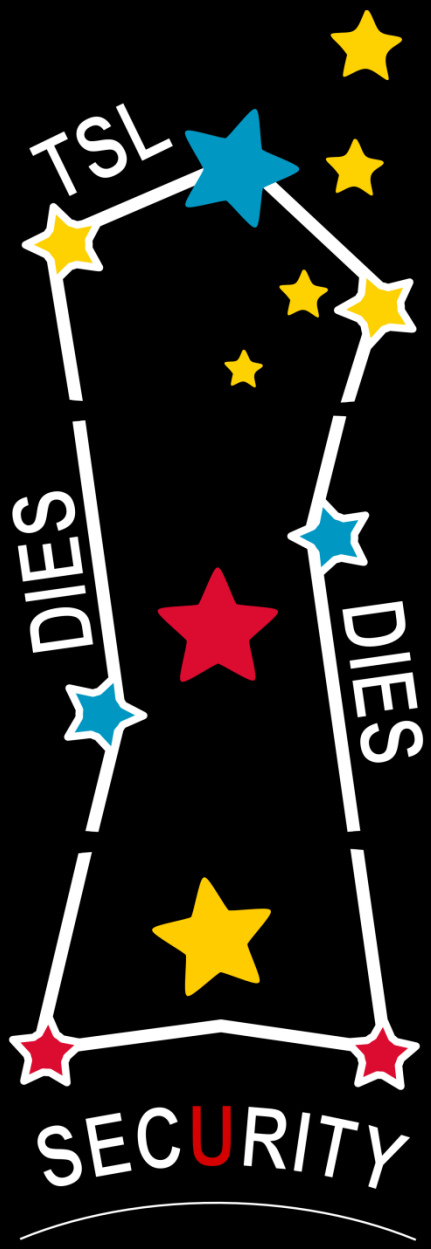
UNIVERSITY OF TWENTE.



A CUCKOO'S EGG IN THE MALWARE NEST

ON-THE-FLY SIGNATURE-LESS MALWARE ANALYSIS, DETECTION AND CONTAINMENT FOR LARGE NETWORKS

CHRISTIAAN SCHADE

TWENTE SECURITY LAB

UNIVERSITY OF TWENTE

THE NETHERLANDS

MALWARE WARS

- In the last half-decade malware has evolved into a business
  - Windows is the most attacked platform, OS X also affected
- Symantec & Co show impressive growing rates
  - Use of polymorphism/packers
  - Malware writers are just better ☺
- Dynamic Malware Analysis (DMA)
  - Malware samples are executed in a **sandbox**
  - Analysis results are used to update AV signatures and "detection models"
  - Anubis, CWSandbox, Malheur, Malnet, etc.

LIMITATIONS OF DMA

- Malware writers implemented several **countermeasures** to avoid/slow down the DMA analysis
  - Runs only when user(s) is actually logged in
  - Waits for a certain time frame before activating (10-15 mins)
  - Checks for virtualization / known registry keys / known IPs

- DMA tools usually perform **post-mortem** analysis → users submit their sample(s) and get a report back
  - Limited support to monitor an internal network and protect endpoints
  - 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?

- DMA tools lack information about the execution context and do not offer real-time protection

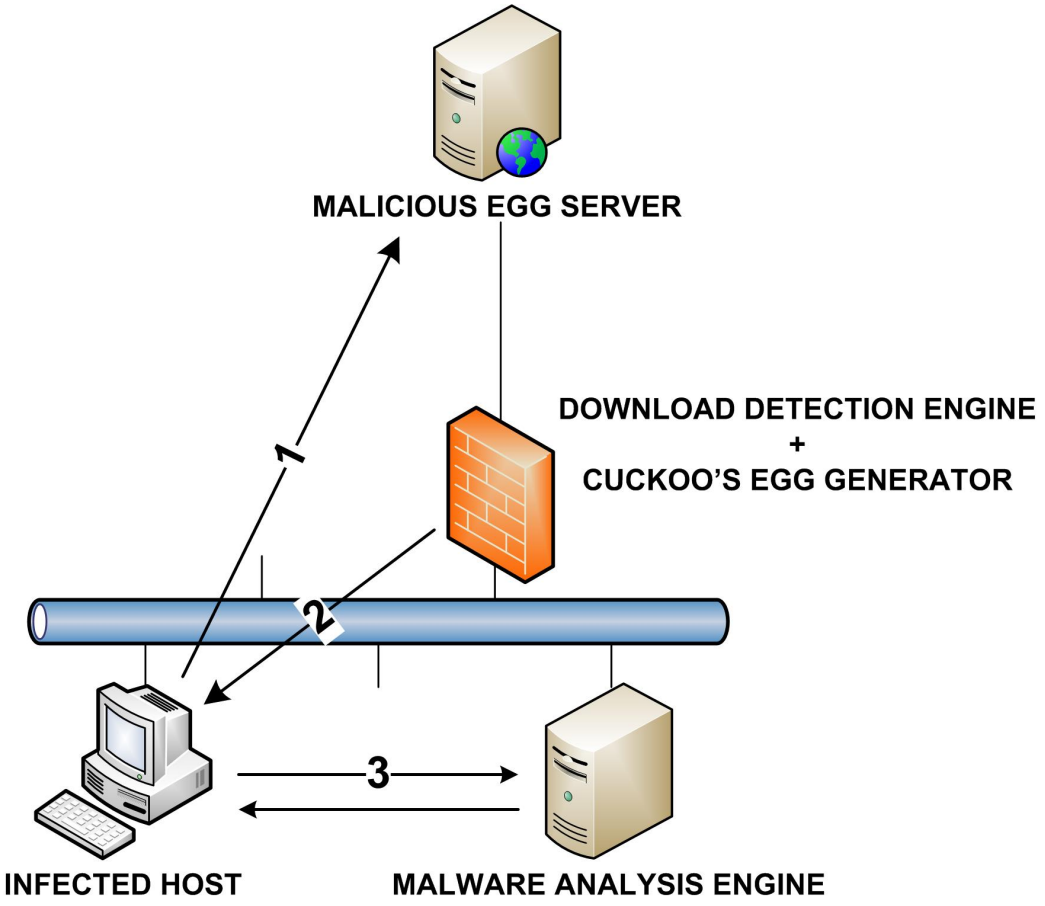
THE IDEA

- ~30% of current malware download additional components once running
  - Require some external “content providers”, usually early compromised servers
  - Content providers might not be online, malware will often need to run several download attempts

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

GENERAL ARCHITECTURE

WE CALL IT AVATAR



LAYING THE EGG…

- We use an algorithm based on TWR to detect “too many” failed attempts, then the egg generator:
  - Checks the requested filename
  - Checks magic numbers in case a file is successfully fetched after several attempts
  - Packs and sends the cuckoo’s egg when # attempts > threshold
- When the egg is executed on the target machine, it attempts to get control over its parent process
  - Depending on the OS version the egg can freeze/terminate the process

…AND PARASITE!

- The egg collects several information about the parent process:
  - Path to the exe
  - Any module that was loaded (full module paths)
  - Window (if any is attached) information: handle, size, caption text
  - Executable size
- The collected information are sent to the MAE, which can stop the egg or perform deeper analysis
  - The egg can send back to the MAE the original parent executable

LIMITATIONS TO OUR APPROACH

- 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 → keys could be disclosed
- Malware writers could use steganography to hide executables into other file formats (e.g., JPEG, like the recent Duqu)
- Malware could leverage the CreateRemoteThread function to execute its code into another process

TESTS

- Avatar 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 – PoC
  - ~10 malware families, huge collection (almost) publicly available from the authors of Malheur (2009) → 75 samples
- Dataset B – evaluation of false positives/negatives
  - everyday malware we received in our mailboxes during a week time → 30 samples + 30 benign samples

TEST RESULTS – DATASET A

Malware family	# of samples	# of samples marked as malicious by the DDE	# samples that executed the cuckoo's egg
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	Correctly identified by the DDE	28/30
	That executed the cuckoo's egg	27/30 (27/28)
	Correctly identified as malware by heuristics	13/30 (13/27)
	Erroneously identified as goodware by heuristics	2/30 (2/27)
	Sent to the MAE for analysis	12/30 (12/27)
Goodware	Erroneously identified by the DDE	10/30
	Correctly identified as goodware by heuristics	6/30 (6/10)
	Erroneously identified as malware by heuristics	2/30 (2/10)
	Sent to the MAE for analysis	2/30 (2/10)

CONCLUSION

- Avatar raises the bar of malware analysis
  - No software is required to run at the endpoint
  - 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 circumvented, but this will also make it more difficult for malware writers
  - No countermeasure has been observed so far in our tests

DEMO

QUESTIONS

?