UBER: Combating Sandbox Evasion via User Behavior Emulators

Pengbin Feng, Jianhua Sun, Songsong Liu, Kun Sun

GEORGE MASON UNIVERSITY

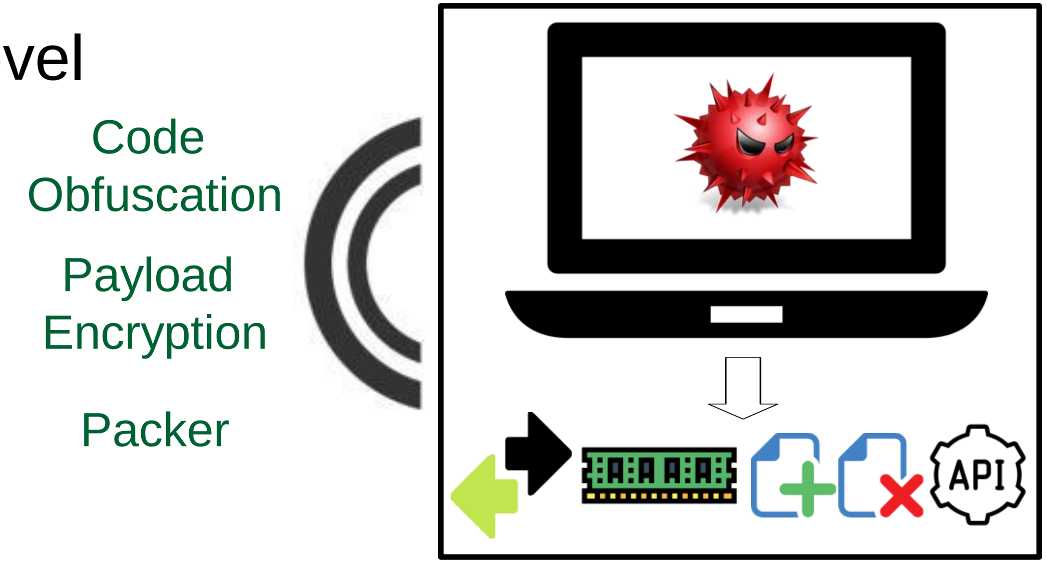
WILLIAM & MARY
CHARTERED 1693

Outline

- Background & Motivation
- System Design
- Implementation & Experiment
- Discussion & Future Work
- Conclusion

Background: Malware Analysis

- Static analysis
  - Decompile program to check risky patterns
  - Analyze all possible code path, relatively fast
  - Cannot handle code obfuscation techniques
- Sandbox-based analysis
  - Monitor runtime behaviors at various level
  - The ability to handle code obfuscation
  - Widely used in cyber security teams

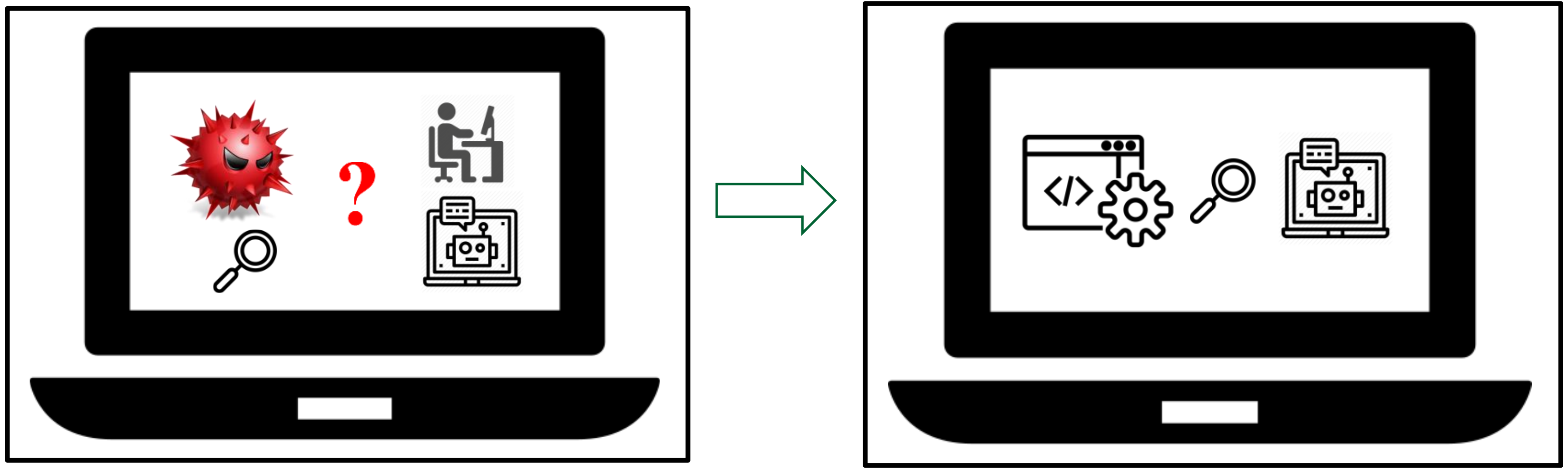

Code Obfuscation

Payload Encryption

Packer

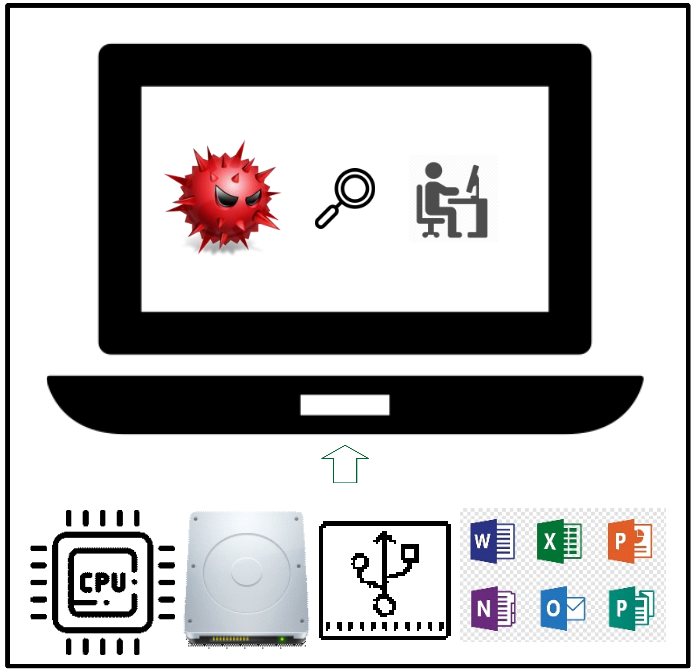
Background: Anti-Sandbox Techniques

- Evasion techniques to circumvent sandbox
  - Malware alters its behaviors when detecting sandbox environment
  - Include detect indicators, such as system setting[1], analysis instrumentation module or drivers[2], user-like mouse clicking[3], as well as time attacking[4], CPU virtualization[5], etc.
  - Evolve from simple environment-specific configuration detection to complex user behavior detection



Background: Anti-Anti-Sandbox

- Multiple mitigation strategies [6] to defeat anti-sandbox
  - State modification: modify the execution state at given points to force code to take alternative branches
  - Multi-platform record & replay: record malware execution information and replay execution code from multiple platforms
  - Bare metal analysis: directly perform instrument analysis on physical machine
  - Hide environmental artifacts through hook function
  - All strategies try to ensure realistic configuration for sandbox environment

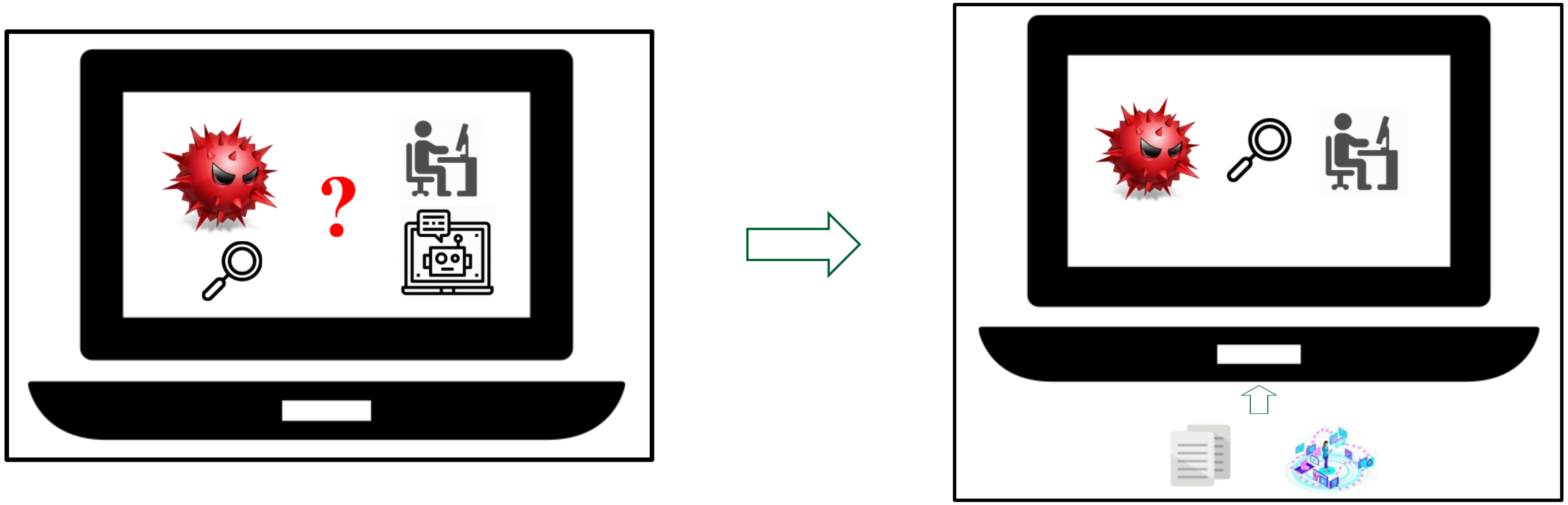


Definition: Usage Artifacts Analysis

- Existing strategies are ineffective in countering **usage artifacts analysis** [7] based sandbox evasion
- Usage artifacts analysis
  - In real system, normal usage contains various actions like browsing website, editing office word, etc. leading to a variety of **artifacts**
  - In sandbox environment, running specific analysis software and lacking abundant functions, leading to little artifacts
  - **Artifacts**: files/traces: Temporary Files, DNS, Bookmarks, Cookies, Log Entries, etc. as a results of accumulation normal usage
  - **Usage artifacts analysis**: Identifying usage artifacts generated by normal user activities to distinguish sandbox from real system

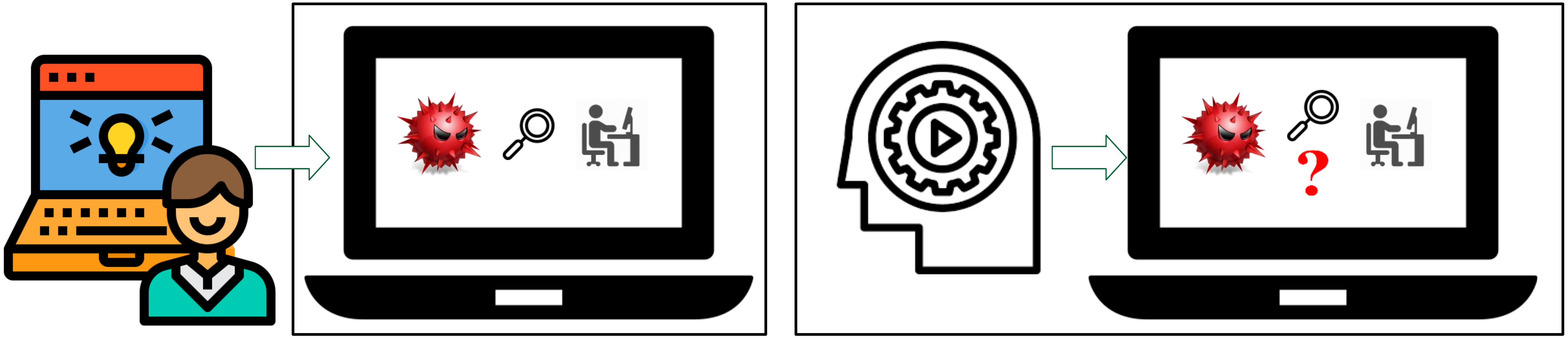
Motivation: Defeat Usage Artifacts Analysis

- Tackle the drawback of lacking historical usage artifacts in existing sandbox environment
- Deceive malware a real usage environment
- How to tackle?



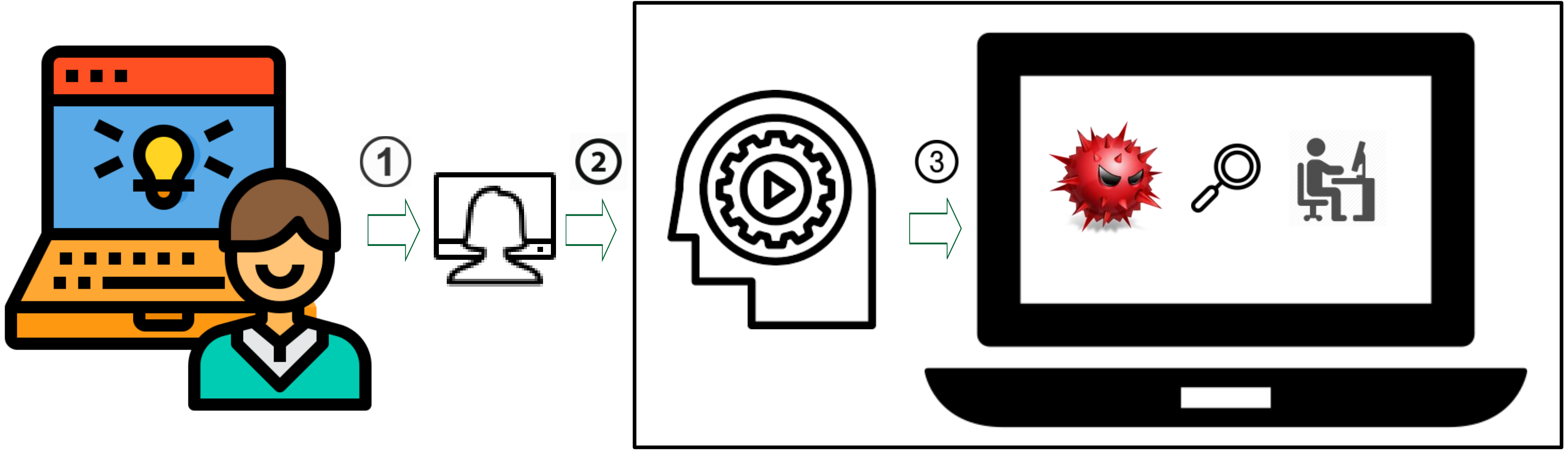
Motivation: Defeat Usage Artifacts Analysis

- Two potential solution
  - Option 1: Clone real user system
    - Directly clone real user system to sandbox
    - Privacy violation, artifacts outdating after a period of time
  - Option 2: Simulate user behavior
    - Directly simulate user behaviors in sandbox environments
    - No privacy, how to ensure realistic of artifacts is a great challenge?



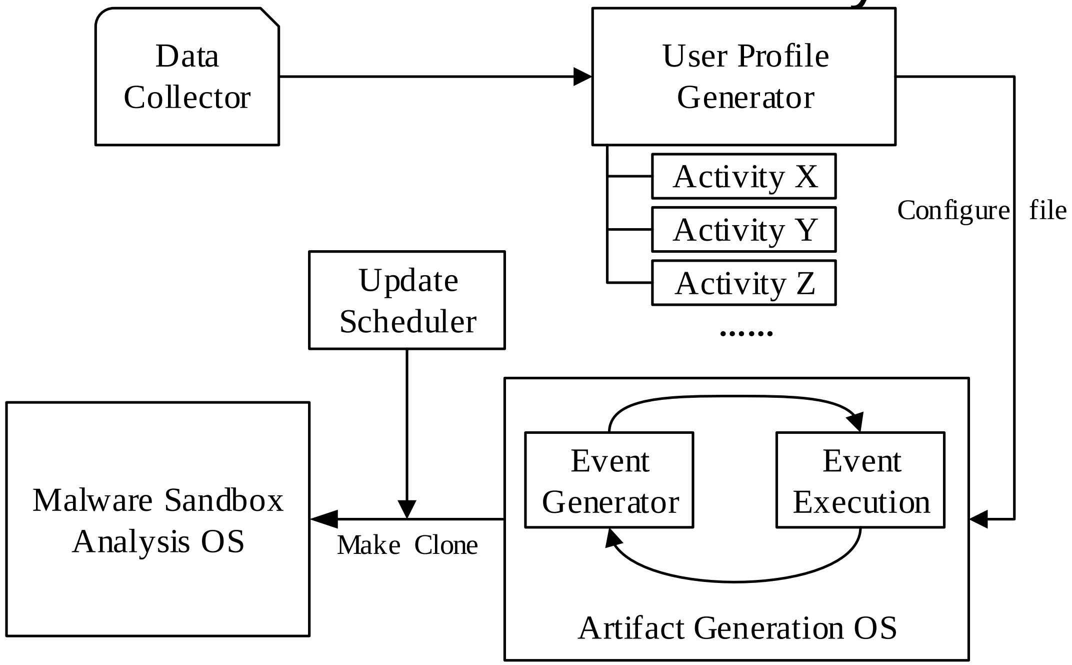
System Design

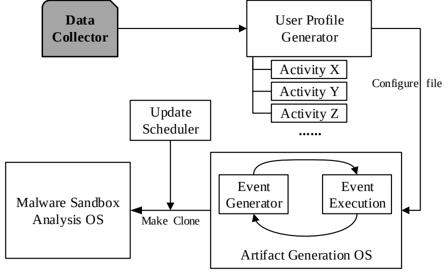
- **User Behavior Emulator (UBER)**
  - Apply the predefined user profile to generate realistic user activities
  - Step 1: collect user data to abstract user behavior profile
  - Step 2: take this profile as input to simulate user behavior
  - Step 3: analyze malware on sandbox environment with generated artifacts



System Architecture

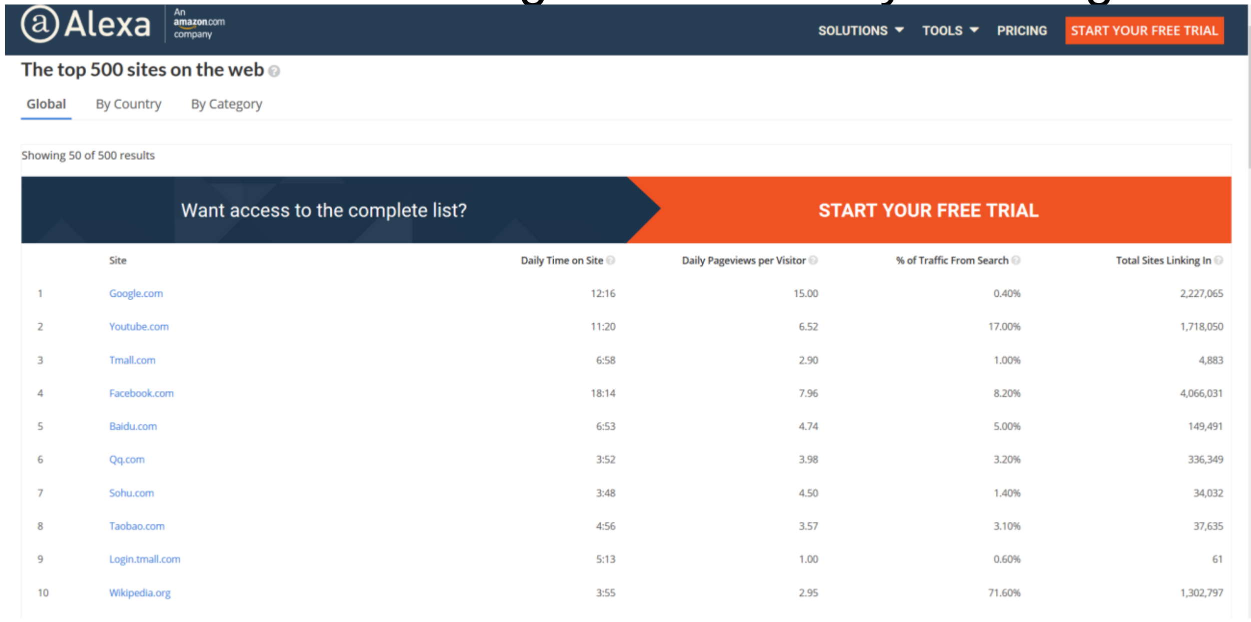
- UBER Overview
  - Gather raw user data which characterizes user behavior
  - Perform statistical and correlation analysis to generate user profile
  - Event Generator create events following user profile and executes them via the Event Execution, resulting in “real” artifacts.
  - Clone to create the malware sandbox analysis environment, keep up-to-date





Data Collector

- Gather information to derive user profile
  - Record application usage time through tracker software
  - Categories application into predefined type
  - Collect public data to build typically operation of activity type
    - Alexa: most frequently visited websites
    - Google Trends: daily trending items



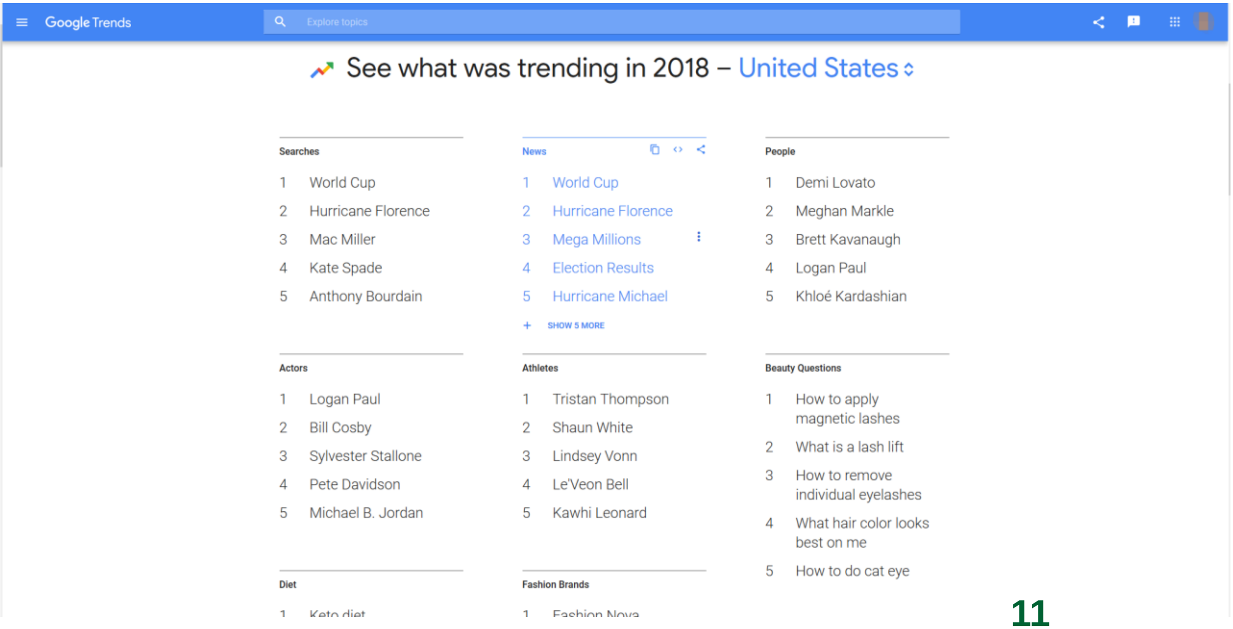
The top 500 sites on the web

Global By Country By Category

Showing 50 of 500 results

Want access to the complete list? START YOUR FREE TRIAL

	Site	Daily Time on Site	Daily Pageviews per Visitor	% of Traffic From Search	Total Sites Linking In
1	Google.com	12:16	15.00	0.40%	2,227,065
2	Youtube.com	11:20	6.52	17.00%	1,718,050
3	Tmall.com	6:58	2.90	1.00%	4,883
4	Facebook.com	18:14	7.96	8.20%	4,066,031
5	Baidu.com	6:53	4.74	5.00%	149,491
6	Qq.com	3:52	3.98	3.20%	336,349
7	Sohu.com	3:48	4.50	1.40%	34,032
8	Taobao.com	4:56	3.57	3.10%	37,635
9	Login.tmall.com	5:13	1.00	0.60%	61
10	Wikipedia.org	3:55	2.95	71.60%	1,302,797



See what was trending in 2018 – United States

Searches
1 World Cup
2 Hurricane Florence
3 Mac Miller
4 Kate Spade
5 Anthony Bourdain

News
1 World Cup
2 Hurricane Florence
3 Mega Millions
4 Election Results
5 Hurricane Michael
+ SHOW 5 MORE

People
1 Demi Lovato
2 Meghan Markle
3 Brett Kavanaugh
4 Logan Paul
5 Khloé Kardashian

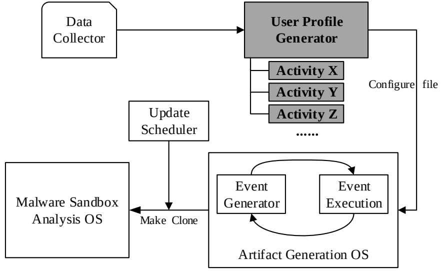
Actors
1 Logan Paul
2 Bill Cosby
3 Sylvester Stallone
4 Pete Davidson
5 Michael B. Jordan

Athletes
1 Tristan Thompson
2 Shaun White
3 Lindsey Vonn
4 Le'Veon Bell
5 Kawhi Leonard

Beauty Questions
1 How to apply magnetic lashes
2 What is a lash lift
3 How to remove individual eyelashes
4 What hair color looks best on me
5 How to do cat eye

Diet
1 Keto diet

Fashion Brands
1 Fashion Nova

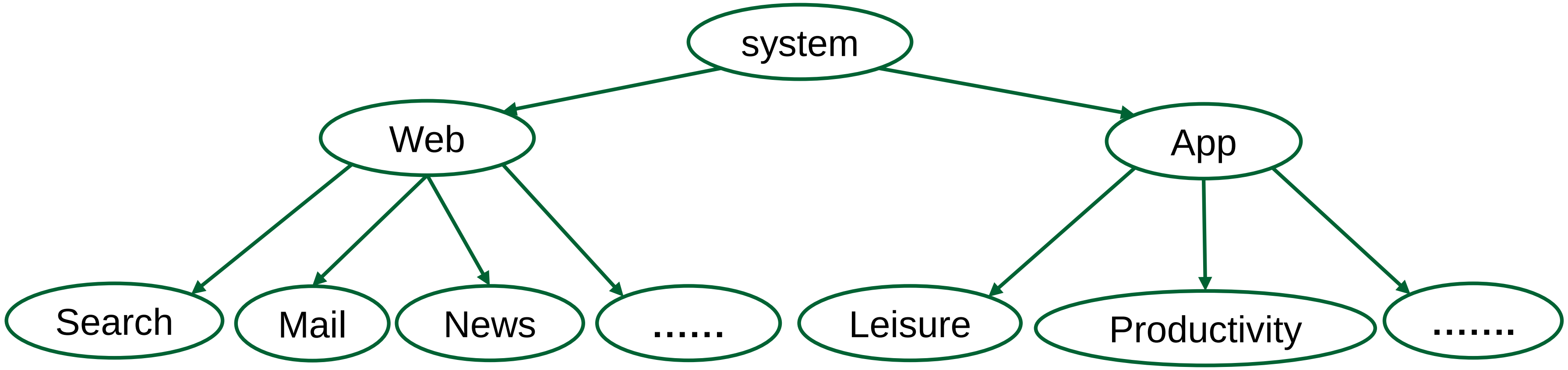


User Profile Generator

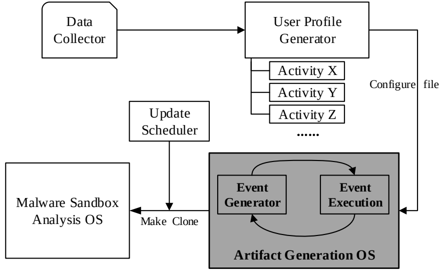
- Statistical analysis of collected information
- Output configuration file defining how to perform user actions
- An brief example of user profile
    - Duration: average computer usage time
    - Probability, likelihood a user would perform specific activities
    - Predefined type: usage experience

```
# System usage (Start time, Duration)
onTimes: 0800+0100-0100, 210
onTimes: 1300+0030-0030, 270

# Activity type of user (Type, Probability)
ActivityTypes: web, 60|app, 40
```

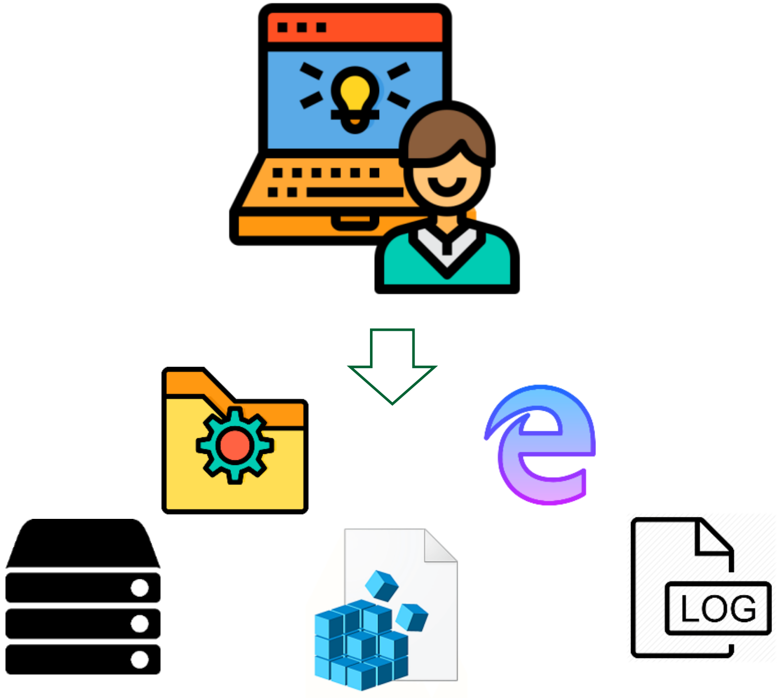


Artifact Generation OS

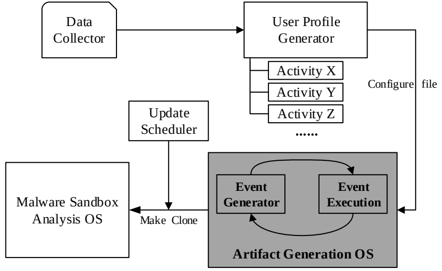


- Typical system artifacts
  - Accumulation from normal usage with various actions
  - Indicate historical usage
  - Existing big difference between sandbox and real system

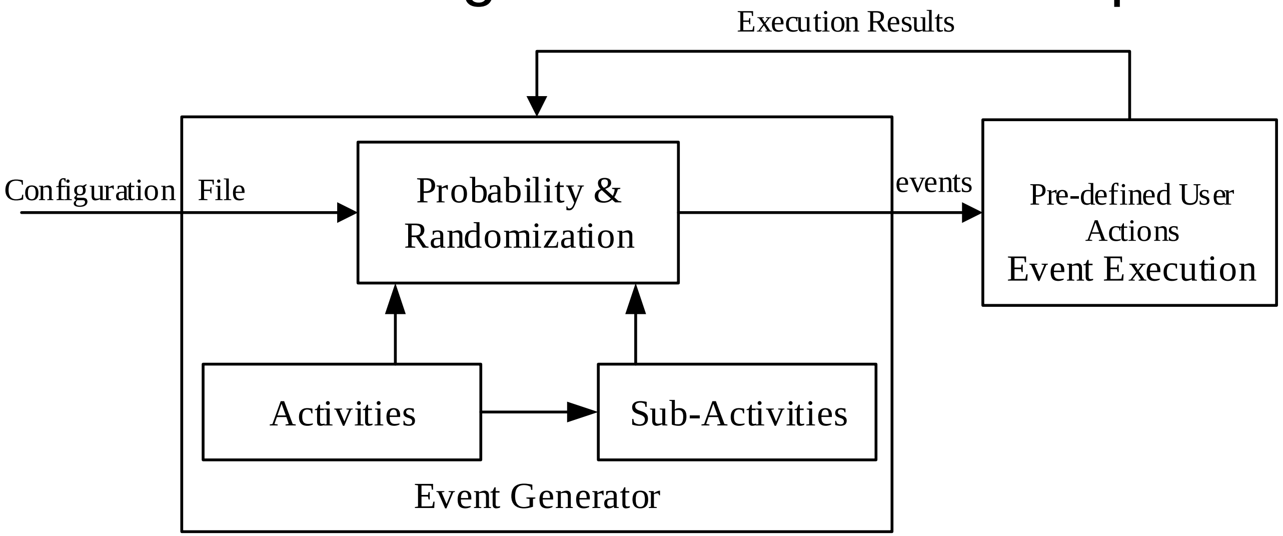
File System	Downloaded Files
Browser	Total URLs Visited, Unique Domains, Cookies, Bookmarks, Temporary Internet Files
Network	ARP Entries, DNS Records, Bytes Sent, Active Connections
Registry	MUI Cache, Userassist Entries, MRU Entries, Registry Size
System	System Log Entries, Application Log Entries

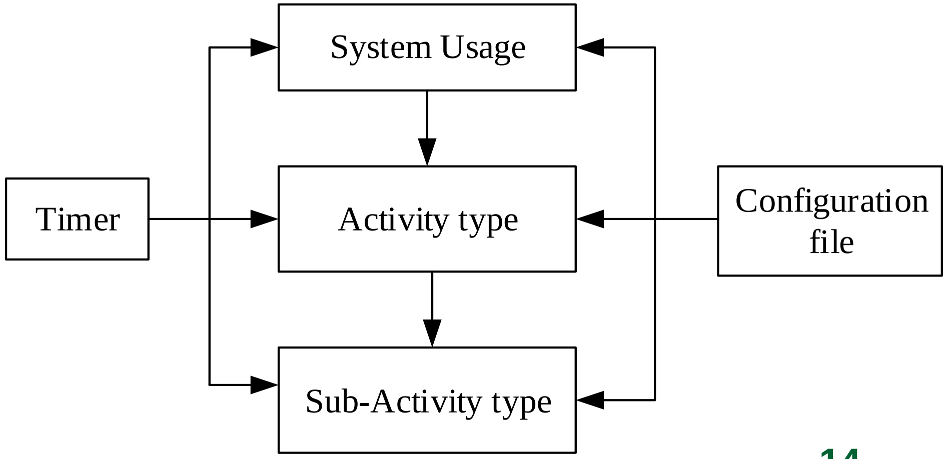


Artifact Generation OS

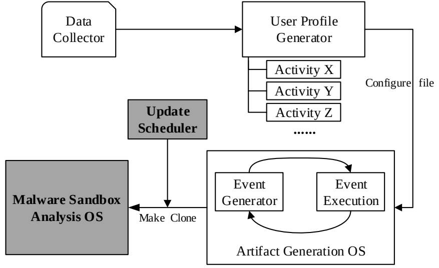


- Event Generator
  - Make decision on which events will be performed
  - The P & R function takes the configuration file to select the activities and the corresponding sub-activities
  - The timer ensure the emulation time not exceed limits in configuration
- Event Execution
  - Executing the events based on predefined actions





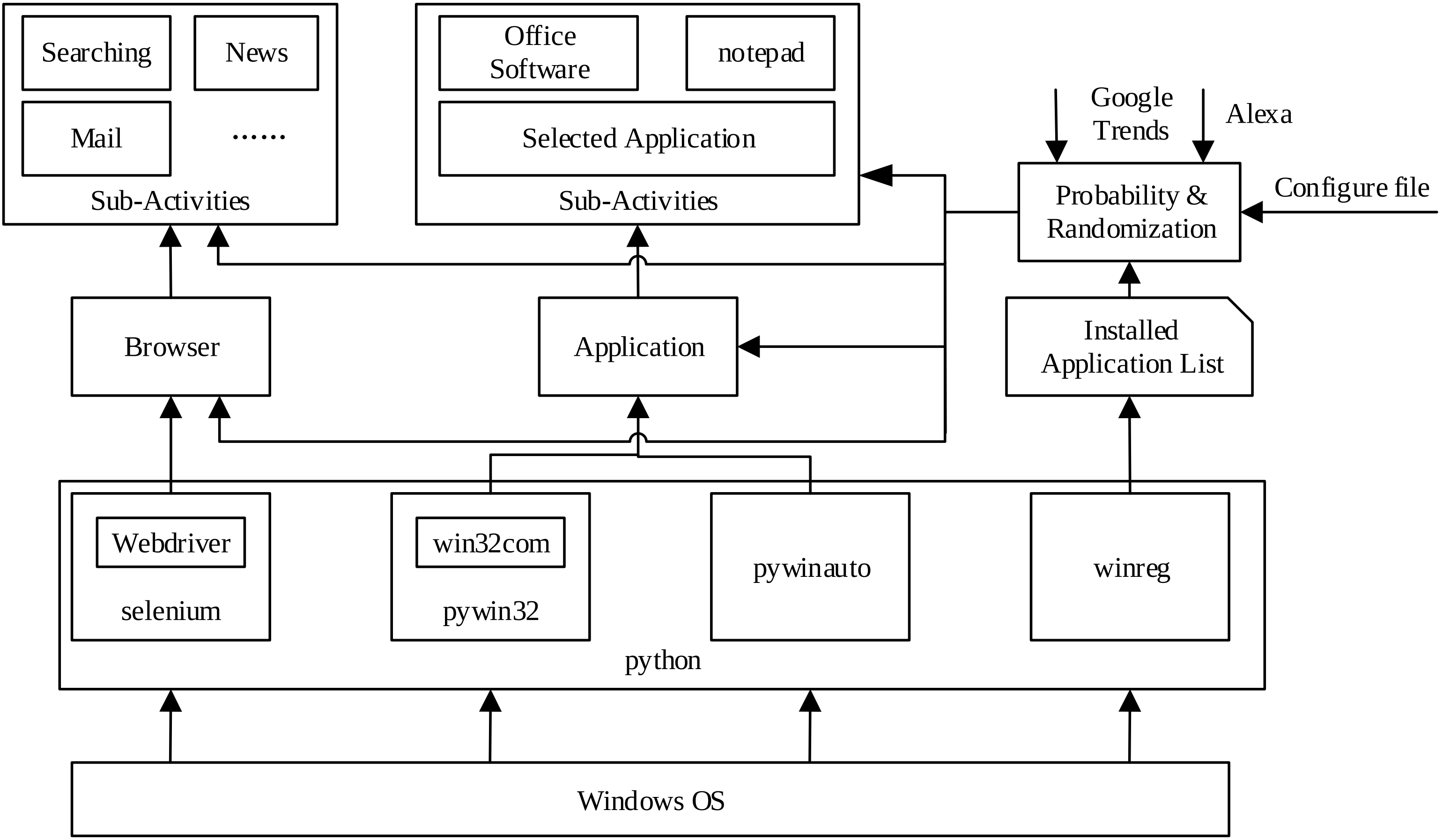
Malware Analysis OS



- Malware Sandbox Analysis OS
  - Execute malware and gather runtime information
  - The emulation software should not be executed on this OS
  - Avoid runtime resource competition between emulation and malware
  - Reduce the chance of malware identifying sandbox through detecting the emulation driver
- Update Scheduler
  - Create copy of Artifact Generation OS to sandbox analysis
  - Regularly copy to keep the artifacts of malware sandbox up-to-date

Implementation

- Implement a prototype through python scripts
  - Use python module *Selenium*, *Pywin32* and *Pywinauto* to control the browser and application
  - Recruit several volunteers to generalize user profile
  - Perform UI interaction in human-like speed
  - Perform activities in human-like habits
  - Manually parse commonly accessed websites and GUI elements from popular applications



Experiment

- Implement automation script with NirSoft[1] to collect artifacts
- Collect artifacts from multiple available sandbox systems and real user systems
- Artifacts Difference

Artifacts	Sandbox	Real Systems	**Difference**
Downloaded Files	0	27	**27**
Total URLs Visited	3	301	**298**
Unique Domains	0	55	**54**
Cookies	0	71	**71**
Bookmarks	0	310	**310**
Temporary Internet Files	0	921	**44**
Bytes Sent	2731035	43007337	**40276302**
MUI Cache	2	211	**209**
Userassist Entries	33	62	**29**
MRU Entries	57	433	**376**
Registry Size	52521688	73218690	**20697002**
System Log Entries	774	1715	**841**
Application Log Entries	293	1290	**997**

[1] https://www.nirsoft.net/

Experiment

- Experiment Platform
  - Host System: Ubuntu 18.04 LTS, Intel Xeon(R) E5-2620 CPU @ 2.40GHz x 12 and 16 GB
  - VMs: deploy VirtualBox with 3 vCPUs and 4GB memory
- Measurement Effectiveness
  - Baseline: VMs with fresh installed Oses
  - Baseline + User Operation: Manually operate cloned VMs as “Real”
  - Baseline + UBER: Deploy UBER on these VMs as “Sandbox”

Experiment

- Measurement
  - After one month, the two systems accumulate similar comparable amount of artifacts

Artifacts	Baseline	Baseline + User Operation	Baseline + UBER
Downloaded Files	0	27	34
Total URLs Visited	3	**1786**	**1766**
Unique Domains	1	**373**	**354**
Cookies	5	31	55
Bookmarks	0	**151**	**164**
Temporary Internet Files	19	**57**	**55**
Bytes Sent	2124684	**5225592**	**5012932**
Active Connections	6	**50**	**46**
MUI Cache	14	**26**	**24**
Userassist Entries	43	**73**	**74**
MRU Entries	17	**128**	**136**
Registry Size	87030444	92026650	91356255
System Log Entries	813	845	921
Application Log Entries	694	1124	1208

Realistic Artifacts

Discussion & Future work

- UBER is a complementary to existing mitigation solution
- Data Collection
  - Malware targets specific individuals or organizations
  - Defining the profile of specific individuals
- Software Specific Artifacts
  - UBER emulates popular software, lacks artifacts of specific software
  - Modify UBER to emulate this software to generate unique artifacts
- Validation of Artifacts
  - Check the content of artifacts (e.g., correctness of documents)
  - Plan to integrate fake document generation methods FORGE [8] into UBER

Conclusion

- Perform the study of malware sandbox evasion techniques that leverage system artifacts analysis
- Propose UBER, which generate realistic usage artifacts based on the predefined user profile
- Implement a prototype, and verify its effectiveness through experiments

Questions!

Reference

[1] Lindorfer, Martina, Clemens Kolbitsch, and Paolo Milani Comparetti. "Detecting environment-sensitive malware." International Workshop on Recent Advances in Intrusion Detection. Springer, Berlin, Heidelberg, 2011.

[2] Chen, Xu, et al. "Towards an understanding of anti-virtualization and anti-debugging behavior in modern malware." 2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN). IEEE, 2008.

[3] Keragala, Dilshan. "Detecting malware and sandbox evasion techniques." SANS Institute InfoSec Reading Room 16 (2016).

[4] Brengel, Michael, Michael Backes, and Christian Rossow. "Detecting hardware-assisted virtualization." International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Springer, Cham, 2016.

[5] Alwabel, Abdulla, et al. "Safe and automated live malware experimentation on public testbeds." 7th Workshop on Cyber Security Experimentation and Test ({CSET} 14). 2014.

[6] Bulazel, Alexei, and Bülent Yener. "A survey on automated dynamic malware analysis evasion and counter-evasion: PC, mobile, and web." Proceedings of the 1st Reversing and Offensive-oriented Trends Symposium. ACM, 2017.

[7] Miramirkhani, Najmeh, et al. "Spotless sandboxes: Evading malware analysis systems using wear-and-tear artifacts." 2017 IEEE Symposium on Security and Privacy (SP). IEEE, 2017.

[8] Chakraborty, Tanmoy, et al. "FORGE: A Fake Online Repository Generation Engine for Cyber Deception." IEEE Transactions on Dependable and Secure Computing (2019).