DefWeb: Defending User Privacy against Cache-based Website Fingerprinting Attacks with Intelligent Noise Injection

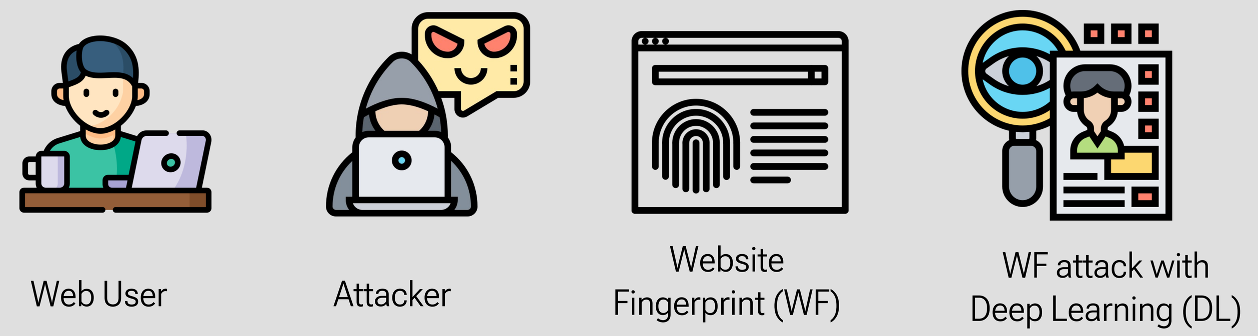
Seonghun Son, Debopriya Roy Dipta, Berk Gulmezoglu

Microarchitecture and Artificial Intelligence Security (MAIS) Laboratory, Iowa State University

Introduction

❖ ***Motivation:***

- ***Issue***
  - Cache-based Website Fingerprinting (WF) attacks violate user privacy by exploiting shared CPU resources, even on Incognito or Tor browsers.
- ***Why it matters?***
  - Existing defense techniques either fail to fully obfuscate data or cause significant performance overhead.
  - Precedent work
    - Oren et al. (2015) [1] : Cache attacks in JavaScript environments with an attack accuracy of 78.4% and mitigation of **76.2%**
    - Shusterman et al. (2019) [2]: Cache occupancy based WF attack, achieving 95.7% accuracy and mitigated to **62.0%** through noise injection.
    - Cook et al. (2022) [3]: Loop-counting based WF attack with an accuracy of 95.7%, which was reduced to **46.2%** using randomized timers.



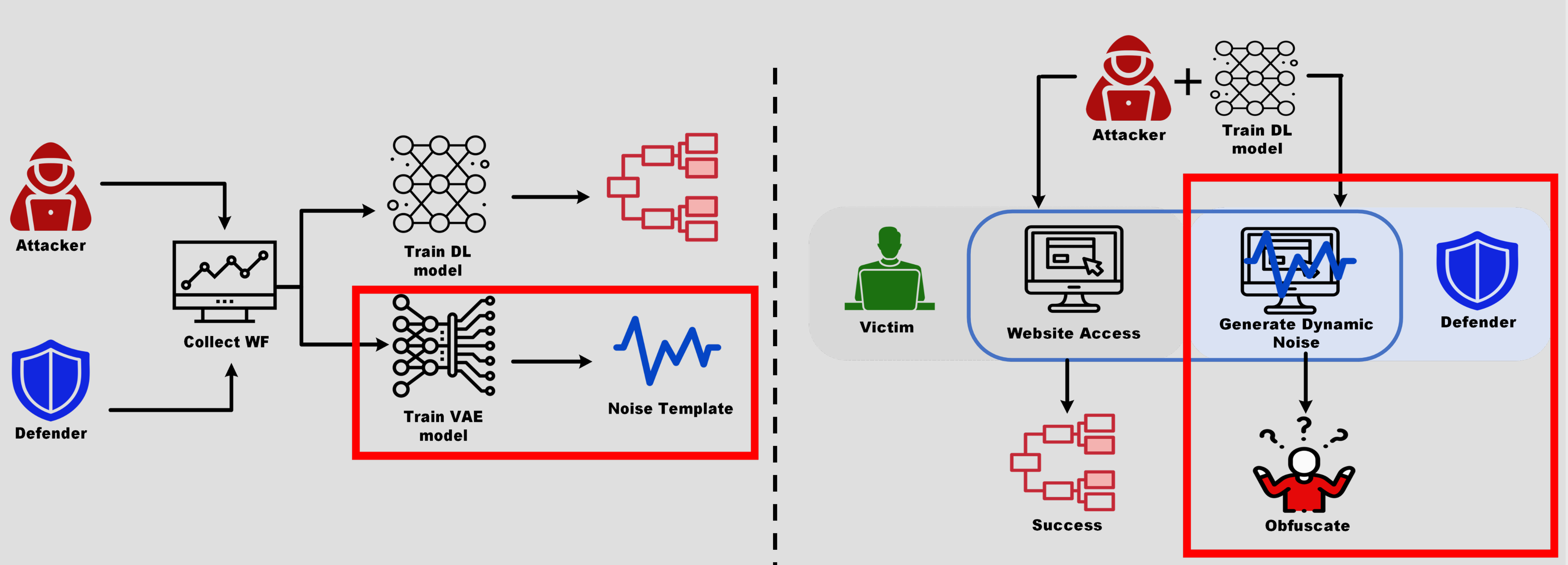
❖ ***Proposed Solution:***

- ***Solution***
  - A novel defense mechanism that injects intelligent noise using a generative learning model to protect user privacy during web browsing activity.
- ***Objective***
  - Decrease the attacker Machine Learning(ML) model's accuracy with minimal performance overhead.

Method

❖ ***Overview***

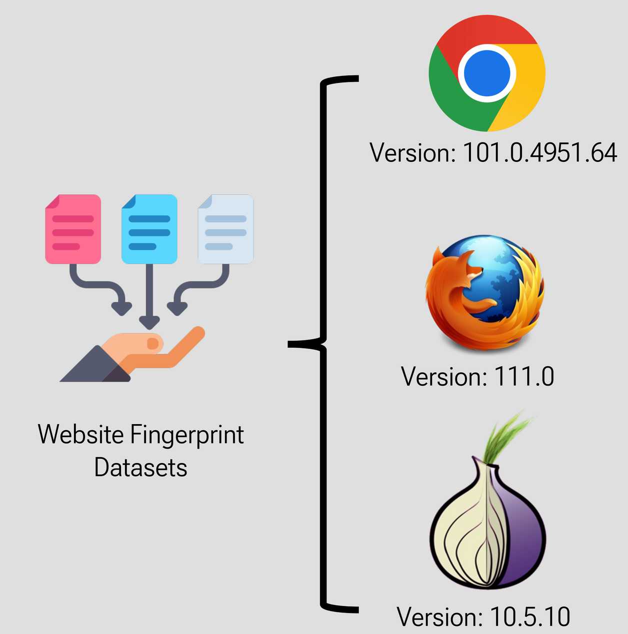
- *DefWeb* employs a dynamic noise injection (noise template) utilizing a generative learning deep learning model (Variational Autoencoder).
- ***Online phase***: Training the defense mechanism by collecting WF data and generating noise templates
- ***Offline phase***: Applying the generated noise in real-time during website browsing to obfuscate the fingerprints and protect user privacy



Offline Phase | Online Phase

❖ ***Data Collection***

- ***Process***: Collect website fingerprints via the cache occupancy channel [2] and loop-counting [3]

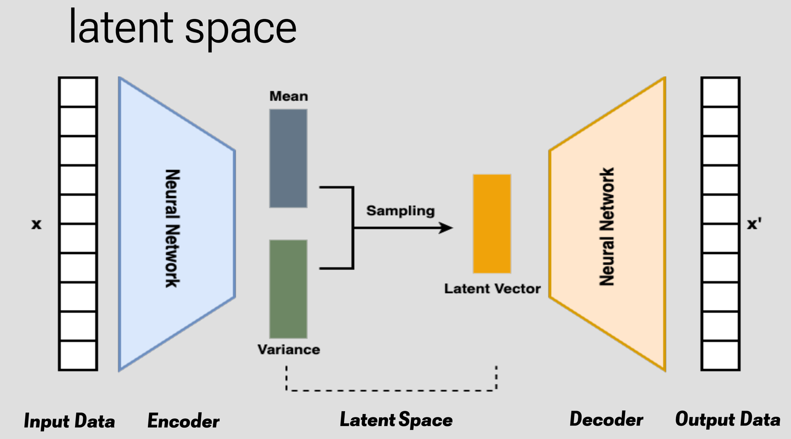


1. 360.cn	21. dailymotion.com	41. imdb.com
2. 9gag.com	22. digikala.com	42. imgur.com
3. abs-cbn.com	23. discord.com	43. indeed.com
4. adobe.com	24. dropbox.com	44. intuit.com
5. airbnb.com	25. ebay.com	45. jd.com
6. aliexpress.com	26. espn.com	46. kompas.com
7. allegro.pl	27. espncricinfo.com	47. linkedin.com
8. amazon.com	28. etsy.com	48. liputan6.com
9. apple.com	29. exoclick.com	49. live.com
10. archive.org	30. flipkart.com	50. mail.ru
11. baidu.com	31. force.com	51. mediafire.com
12. bbc.com	32. foxnews.com	52. medium.com
13. bing.com	33. github.com	53. mozila.org
14. booking.com	34. globo.com	54. msn.com
15. bukalapak.com	35. godaddy.com	55. naver.com
16. canva.com	36. goodreads.com	56. netflix.com
17. chase.com	37. google.com	57. nih.gov
18. craigslist.org	38. healthline.com	58. nordstrom.com
19. csdn.net	39. hulu.com	59. office.com
20. dailymail.co.uk	40. ikea.com	60. okezone.com ......

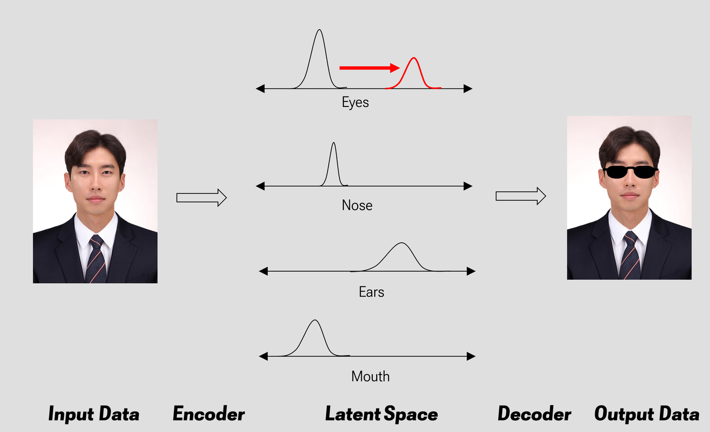
Alexa's Top 150 website list

❖ ***Latent Space Representation Using Variational Autoencoder (VAE)***

- High-dimensional WF datasets to a lower-dimensional latent space utilizing VAE
- ***Objective***: Compress meaningful features and separate WF into clusters in the latent space



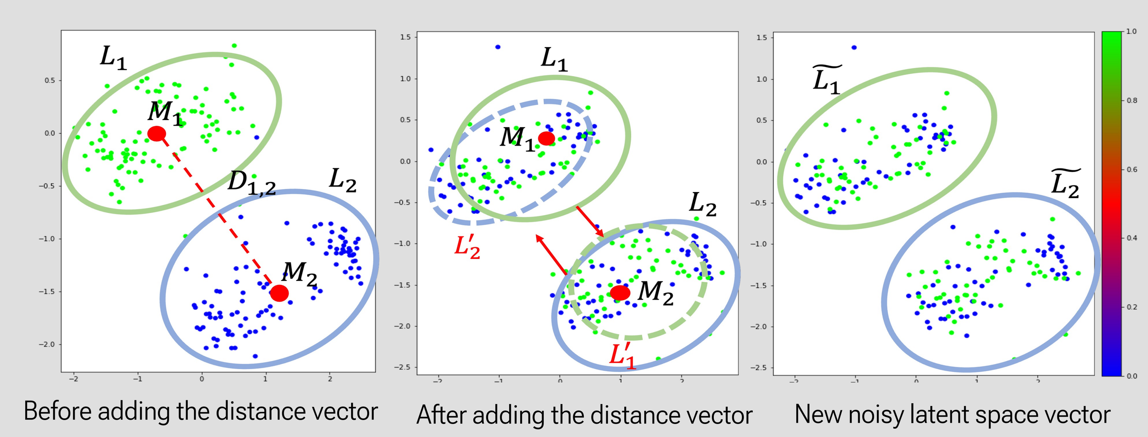
Structure of Variational Autoencoder (VAE)



VAE Example

❖ ***Noise Template Creation***

- Generate minimal noise templates manipulate in the latent space
- ***Process***: Calculate the distance between clusters in the latent space and generate noisy WF datasets to obfuscate the WF



Before adding the distance vector | After adding the distance vector | New noisy latent space vector

$$L_i' = L_i + D_{i,j}$$

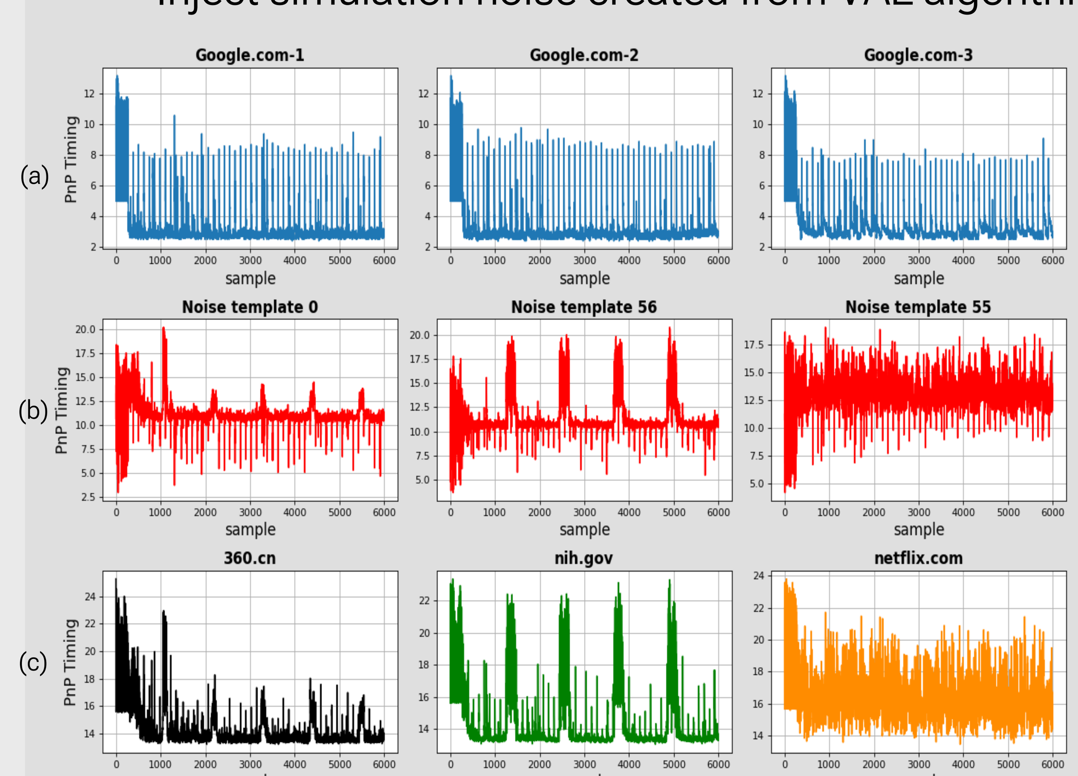
$$S' = V_D(L_i')$$

$$\text{where } i, j = \{0, \cdots, w-1\}$$

Clusters in the latent Space (W=google.com, amazon.com, D=2)

❖ ***Simulation Noise Template Injection***

- Inject simulation noise created from VAE algorithm



(a) Original WF, (b) Noise Template, (c) Noise injected WF

Dim.	Acc.	Re-Con.	Noise dataset	Injected
50	53.8	90.6	6.1	5.2
100	93.8	***94.4***	6.3	4.5
200	97.4	93.7	24.9	23.3
300	97.7	95.6	36.8	33.1

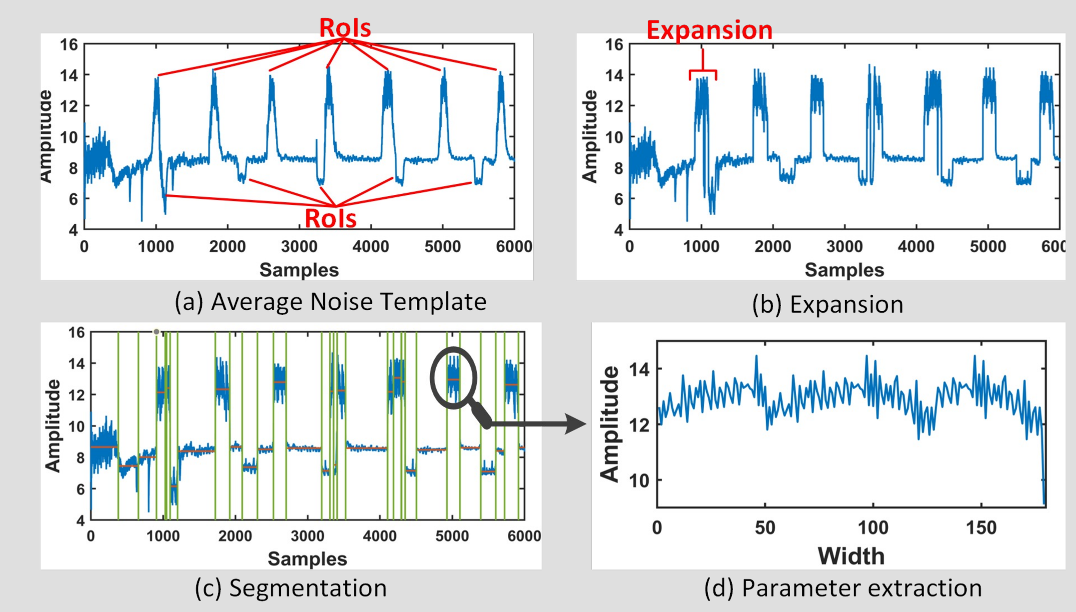
Dimension selection on the accuracy (%)

Browser	CNN			
	C = 2	C = 3	C = 4	C = 5
Chrome	3.9 ± 3.0%	4.8 ± 3.2%	24.6 ± 10.6%	32.7 ± 19.7%
Firefox	1.6 ± 1.0%	1.3 ± 0.8%	5.7 ± 4.1%	16.8 ± 12.7%

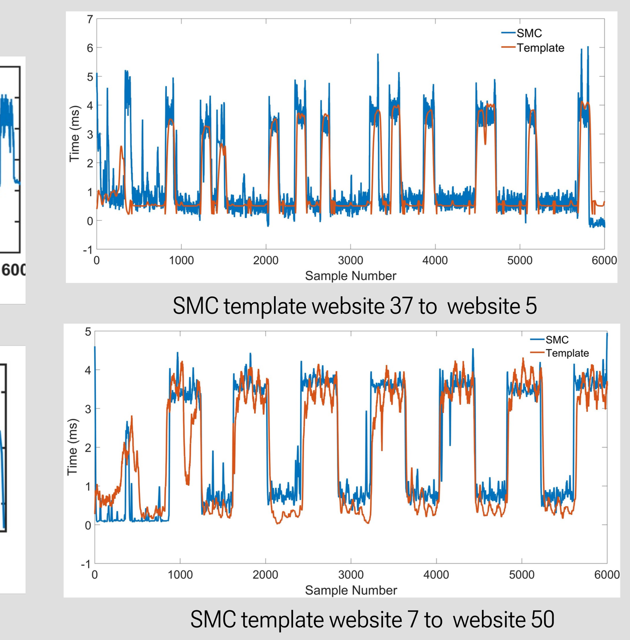
Shift value *c* selection based on accuracy (%)

❖ ***Practical Noise Injection utilizing Self-Modifying Code (SMC)***

- Inject practical noise in microarchitecture during website rendering
- ***Process***:
  - Misalignment
  - Segmentation into Dynamic Noise Block from Practical Noise Template
  - Look-up table creation
  - Practical noise injection in Intel TigerLake microarchitecture



Steps of creating practical noise



Practical noise example

Results

❖ ***Accuracy Degradation***

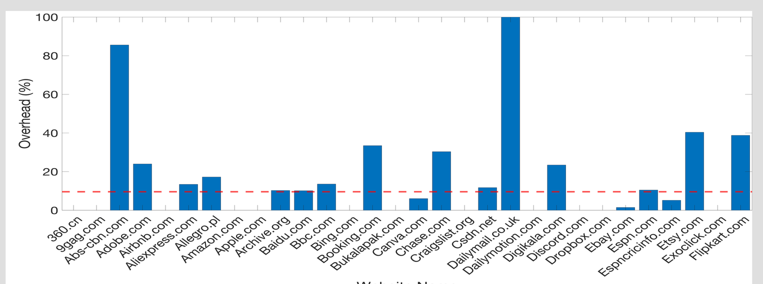
- The classification accuracy for 100 websites drops to 28.8%, 29.7%, and 5.2% accuracy for Chrome, Firefox, and Tor.
- The classification accuracy for 150 websites drops to 24%

Attack	Cache-Sweep	Interrupt Injection	*DefWeb*	
			Chrome & Firefox	Tor
Loop-Counting Attack[4]	x1.03	x1.42	***x3.32***	***x9.2***
Sweep-Counting [32]	x1.03	x1.54	***x3.93***	

WF attack accuracy degradation

❖ ***Performance Overhead***

- Performance overhead tool *WebAPI* and *Selenium* library to measure rendering time.
- It is a better performance tool compared with Benchmarks since we directly check the overhead in a web environment,



Defense technique	Cache Shaping	Interrupt Injection	***DefWeb***
Performance Overhead	51.4 – 71.8%	15.7%	***9.5%***

Performance overhead

Conclusion

❖ ***Future work***

- SMC creation in the browser environment can be used
- The transferability of *DefWeb* can be investigated

❖ ***Conclusion***

- *DefWeb* demonstrates that intelligent noise injection can decrease the attacker Deep learning model's accuracy significantly compared to other method.
- The performance overhead introduced by *DefWeb* is less than other techniques.

❖ ***References***

[1] Yossef Oren, Vasileios P Kemerlis, Simha Sethumadhavan, and Angelos D Keromytis. 2015. The spy in the sandbox: Practical cache attacks in javascript and their implications. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. 1406–1418.
[2] Anatoly Shusterman, Lachlan Kang, Yarden Haskal, Yosef Meltser, Prateek Mittal, Yossi Oren, and Yuval Yarom. 2019. Robust website fingerprinting through the cache occupancy channel. In 28th {USENIX} Security Symposium ({USENIX} Security 19). 639–656.
[3] JackCook,JulesDrean,JonathanBehrens,andMengjiaYan.2022.There'salways a bigger fish: a clarifying analysis of a machine-learning-assisted side-channel attack.. In ISCA. 204–217.