Caching systems: attacks and countermeasures - A Survey

Write the abstract here.

CCS Concepts: • **Do Not Use This Code → Generate the Correct Terms for Your Paper**; *Generate the Correct Terms for Your Paper*; Generate the Correct Terms for Your Paper; Generate the Correct Terms for Your Paper.

Additional Key Words and Phrases: Do, Not, Use, This, Code, Put, the, Correct, Terms, for, Your, Paper

ACM Reference Format:
. 2025. Caching systems: attacks and countermeasures - A Survey. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX)*. ACM, New York, NY, USA, 16 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Responsibility: Radhika

In Multi-Access Edge Computing (MEC) networks, servers are collocated with Base Stations (BSs) at the edge of the network, to reduce latency incurred by end users, such as Mobile Stations (MSs). These servers are not as powerful as the cloud in terms of RAM, CPU or storage. When end users request for content (images, videos etc.), the requests

Author's Contact Information:

Manuscript submitted to ACM

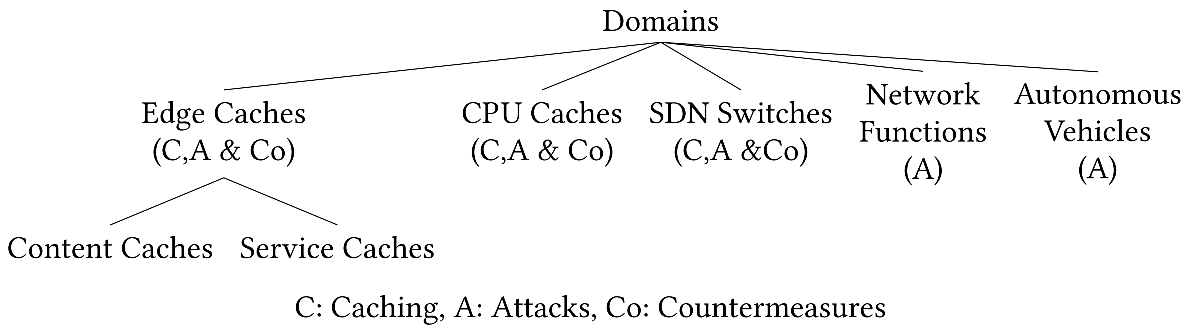


Fig. 1. Domains surveyed

arrive at the edge and are forwarded to the cloud for the content to be downloaded to the end device. Forwarding all requests to the cloud and getting a response can cause high latencies. Moreover, if there are multiple users, the links from the edge to the cloud can get overloaded, causing futher delays. Therefore, each time content is downloaded, it can be cached at the edge.

Services are applications or processes that respond to requests from the user. For example, an Alternate Reality (AR) or Virtual Reality (VR) game requires an MS to send data from the device to the edge, which forwards it to a AR/VR game service running in the cloud. When data is sent such a service, a response is sent to the user and then the user can make the next move. To reduce latency, such a service and its associated database can be downloaded to the edge and cached. This differs from content caching in two ways: 1) A service request requires only a response, which is typically much smaller than content. 2) Content has to be always downloaded to the end device, whereas the service is never downloaded to the end device. 3) The edge can decide when to download a service based on an *admission policy*, while it always downloads content in response to a request.

This survey examines attack methods and corresponding countermeasures for systems that employ caching mechanisms, with particular attention to attacks and defenses relevant to edge caching systems. We do not explore any cryptographic methods, or attacks and countermeasures that are not relevant to edge caching systems. We intend to explore attack methods that may be relevant to caching systems, but the targets of which may not necessarily be caching systems.

Caches are ubiquitous in computer systems. As discussed above, there are caches at the edge, caching content and services. In addition, we explore attacks and countermeasures of two other systems that use caching: CPUs and SDN switches. To explore attacks that are relevant to edge caching, we explore Network Functions and Autonomous Vehicle Simulators. Fig. 1 illustrates the domains we survey.

CPU Caches: In the CPU microarchitecture, L1, L2 and L3 caches reside between the processor and the main memory. Typically, L1 and L2 caches are private to each core in a multiprocessor architecture, whereas L3 is common across cores. These caches are vulnerable to various types of attacks. Expand: [10], [17]

SDN Switches: In computer networks, the forwarding plane or the data plane forwards packets arriving at switches and routers according to the rules installed on them, while the control plane decides which rules to install. In Software Defined Networks (SDN), these planes are separated: the control plane is centralized in a controller, which programmatically controls the forwarding planes present in SDN switches. SDN switches store rules in Ternary Content Accessable Memory (TCAM) and Static Random Access Memory (SRAM). TCAMs are fast, but expensive, while SRAMs are slow and comparatively less expensive. Therefore, TCAMs may be considered caching flow rules. Therefore, the vulnerabilities, attack methods, and countermeasures are directly relevant.

Domain Name Systems: Domain Name Systems (DNS) translate human readable domain names into Internet Protocol (IP) addresses. A DNS cache stores the result of a name lookup at the browser level, at the Operating System (OS) level or at the level of the Internet Service Provider (ISP). Afek et al. [1] provide a history of DNS poisoning attacks.

Content Delivery Networks (CDNs): In a typical content delivery workflow, a user request is first resolved via DNS. The CDN-operated authoritative DNS service selects an appropriate edge server based on factors such as the location of the resolver, network conditions and the load. The DNS response directs the client to a specific CDN edge node, which is often physically colocated within an ISP point of presence, but remains under the operational control of the CDN. The user's HTTP(S) request is then sent directly to this edge server. If the requested content is present in the edge cache, it is served immediately; otherwise, the edge retrieves the content from the origin server (or an upstream parent cache), forwards it to the user, and caches it according to cache-control policies. Commercial agreements between ISPs and CDNs typically govern colocation, connectivity, and traffic locality, while content placement, DNS-based request steering, and caching decisions are centrally managed by the CDN. CDNs cache objects and small state-less functions, unlike an edge cache, which can cache services too. Read [8] for more on this.

Note: Caching in the World Wide Web and Information Centric Networking may also be studied. See [15] and associated papers.

Network Functions: Network Functions are software modules implemented on servers that perform packet processing for reasons other than forwarding or routing, such as securing the network, optimizing traffic, etc. Intrusion Detection and Preventions Systems (IDPS), Firewalls and Wide Area Network (WAN) Optimizers are some examples. It is useful to learn the Attacks and Countermeasures employed in these systems with a view to using them in edge caches.

Autonomous Vehicles: Autonomous Vehicles (AV) aim to revolutionize driving in urban areas and highways. Simulators are used to test the safety of such vehicles. The techniques used to develop a homologation framework (the official certification process that ensures that vehicles comply with government requirements) for critical scenarios for such frameworks could be used for edge caching systems [4]. An AV planner is the part of an autonomous driving system that decides what the vehicle should do next and automated stress testing of planners using generated scenarios could be used for similar purposes for edge caching systems [11].

Our Contributions: We study the domains listed in Fig.1 (and explained above) for ideas relevant to reconnaissance, attacks, and countermeasures for edge service caching systems. Moreover, we propose taxonomies for attacks and countermeasures and summarize findings from the recent literature according to these taxonomies. We also discuss open challenges and future research directions.

We organize prior work along the following axes: attack methods and countermeasures, and domains surveyed. We examine attack methods according to their:

- method of probing (reconnaissance), if applicable or if any and what is probed (replacement algorithm, size of cache, etc.)
- conditions for probing (do they consider network jitter, link delays, link failures, etc.)

- resource consumption of the attack (low, medium, high)
- the extent of knowledge of the system required by the attackers (black box, gray box, or white box)
- relevance to edge service caching
- existence of countermeasures, if any
- verification method (tested on real networks or simulators, whether the datasets used,if any, are realistic)
- limitations (assumptions on the system by the attackers)

We examine countermeasures according to their:

- method (heurisitic, AI etc.)
- impact on performance
- impact on resource consumption,
- tested on real networks or simulators, the datasets used, if any
- relevance to edge service caching
- limitations (are there strong assumptions, are the datasets realistic, etc.)

1.1 Comparison with existing surveys

Responsibility: TBD

1.2 Why this survey is different

Responsibility: TBD

Add comparison with existing surveys on caching systems, Low-rate attacks in SDN,Blackbox attacks etc.

Our contributions:

2 Background and system model

Responsibility: TBD

The domains surveyed in this paper are shown in Fig. 1.

Explain 1) The system model for each domain 2) Definitions of common terms and notations. In the rest of the paper you don't need to define any term again.

2.1 SDN Switches

Responsibility: Robin.

Write briefly about how an SDN works (Switches, Controller, Openflow, TCAM, caching etc.).

2.2 Network Functions

Responsibility: Robin What are Network Functions?

2.3 Relevance to edge service caching

Explain why attacks and countermeasures in domains other than caching are relevant. An example is: The cache itself can be attacked. Attacks can be triggered such that the links from edges to the cloud are saturated. If a TCAM in an SDN switch is considered to be a cache, a cache miss for a packet would result in traffic to the controller until a flow rule is downloaded to the cache.

3 Attack taxonomy

Refer Fig.3 and 4 for the attack taxonomy.

Need to decide which of the ACAs given the figure are low-resource and which are high-resource. Currently, all are grouped under low-resource.

Need to add more survey papers related to attacks on caches (pollution attacks etc.).

4 Attacks and countermeasures

4.1 Bayesian Optimization (High resource attacks)

4.1.1 Introduction to Bayesian Optimization. Responsibility: Vasudeva

This should be 2 pages long and must describe the basic idea with references. Use mathematical notation wherever required.

Bayesian Optimization (BO) is one of the powerful strategies for finding extrema of objective functions. Generally, an optimization problem is also formulated as maximizing function $f(x)$ over a domain $A \subset \mathbb{R}^d$, i.e., $\max_{x \in A \subset \mathbb{R}^d} f(x)$. Particularly, a few characterstics of $f(x)$ make BO the appropriate choice of optimization strategy such as costly to evaluate, unknown/no mathematical representation, no access to derivatives, non-convex, limited observations, noisy observations, etc. [5].

BO is a strategy combining ideas from Bayesian inference (Bayes' theorem), Surrogate modeling (mean function, covariance/kernel function), Sequential decision-making (acquisition functions like expected improvement).

BO is grounded in Bayes' theorem. Bayes' theorem broadly states *posterior* probability of a model (or hypothesis) f_* given dataset (or observations) D equals *likelihood* of f_* given D multiplied by *prior* probability of f_* and divided by the probability of D, i.e.,

$$p(f_* \mid D) = \frac{p(D \mid f_*)\, p(f_*)}{p(D)}$$

since $p(D)$ does not depend on f_*, it is also represented as $p(f_* \mid D) \propto p(D \mid f_*)\, p(f_*)$ in BO literature. Along the lines of Bayes' theorem, BO assumes a prior belief for the unknown objective function f, uses the initial samples in D to fit $f*$ (referred to as *posterior*) as a candidate for f. In order to efficiently sample more data points, BO uses acquisition functions to validate the search space of x_t to acquire a new sample, evaluate it for y_t using f, and refit $f*$, making the new f_* (*posterior*) closer to f. BO continous to sample more data until a stopping criteria. The f is modeled as a random function drawn from a probability distribution over functions. The prior belief in BO represents the space of possible objective functions and is an inductive bias encoding assumptions about the nature of f, such as smoothness, continuity, noise, etc, that make some possible functions more plausible. The prior belief also includes the choice of kernel function and its hyper-parameters, the choice of acquisition function and its hyper-parameters, the choice of other hyper-parameters, among others. For this reason, the prior affects the sampling efficiency in acquisition functions, and overall convergence in BO and thus the choice of prior is crucial.

does GP assume lipschitz-continuous? BO conditions on a dataset D with initial samples, where D is a set of initial i pairs of x, y and $x \in \mathbb{R}^d$, $y \in \mathbb{R}$, i.e., $D = \{x_{1:i}, y_{1:i}\}$ and f evaluates x to y. A prior belief, also referred to as a surrogate model, is assumed based on characteristics of the objective function. In BO literature, the standard surrogate model for f is a Gaussian Process (GP) defined in the Eq. below. A GP is intuitively understood as a distribution over functions; that is, each sample drawn from a GP corresponds to a mathematical function. The mean function and the kernel function (also referred to as the pair-wise covariance function of samples in D) parameterize the characteristics of this distribution. If $x_t \in \mathbb{R}^d$, the GP is d-dimensional, and a 0-dimensional GP can be understood as a Gaussian distribution. See an example BO run in the figure ref2. Examples of other surrogate models are Student-t Process, etc.

$$f_* \sim \mathcal{GP}\big(\mu(x), k(x, x')\big)$$

where $\mu(x)$ represents mean function and $k(x, x')$ is kernel function of the GP. Applying Bayes' theorem on GP leads to inference rules for mean function ($\mu_t(x)$) and kernel function (σ_t^2) after t samples are appended to the initial i in D. Each refit as shown in Eq. (1) increases the probability that a sample f_*, drawn from the GP, is close to the actual f. Generally, the $\mu = 0$ and the $k(x, x) = 1$ at the start. A few choices for kernel function include Squared Exponential (RBF) kernel, Matérn kernel, Rational Quadratic kernel, etc.

$$f_* = p\big(f(x) \mid \mathcal{D}_t\big) \tag{1}$$

The kernel matrix K, calculated using D, and y represent a fit of f_*. At the start f_* is fit using the initial i samples in D (the i is a hyper-parameter and is problem-specific). The f_* is refit with the new sample acquired after a few candidates are collected, validated by an acquisition function, evaluated using the actual objective function f, and finally, appended to the D, see Eqs. (2)-(5).

$$\mu_t(x_{t_c}) = \mathbf{k}_*(x_t)^\top \big(\mathbf{K}_{t-1}\big)^{-1} \mathbf{y}_{1:i+t-1} \tag{2.1}$$

Where $k_*(x_t) \in \mathbb{R}^{i+t-1}$ represents the covariance vector of x_t with the other $i + t - 1$ samples in D, $K_{t-1} \in \mathbb{R}^{i+t-1 \times i+t-1}$ represents the pair-wise covariance matrix of the other $i + t - 1$ samples, and $y_{1:i+t-1} \in \mathbb{R}^{i+t-1}$ represents the y values vector of all $i + t - 1$ samples in D. Intuitively, the pairwise covariance of the $x1 : i + t - 1$ data points is projected onto the y space and weighted by the covariance vector of the new point x_t.

$$\sigma_t^2(x_t) = k(x_t, x_t) - \mathbf{k}_*(x)^\top \big(\mathbf{K}_{t-1}\big)^{-1} \mathbf{k}_*(x) \tag{2.2}$$

Similar to Eq. 2.1 Eq. (2.1), the second-term represents information gain based on kernel function values subtracted from variance $k(x, x)$.

Using the Eqs. (2.1) and (2.2), the mean function and kernel function inferred based on the *posterior* f_* for a candidate x_{t_c} of acquisition function α. Generally, candidates are sampled across the search space of x, and gradient based optimization techniques like L-BFGS-B are used to estimate the maxima as x_t.

$$x_{t+1} = \arg\max_{x \in \mathcal{X}} \alpha(x \mid \mathcal{D}_t) \tag{3}$$

The Eq. 3 is the acquisition function α to efficiently select the next sample x_t to evaluate f for y_t. An ideal choice of α should efficiently balance exploration and exploitation. A few choices for the acquisition function include Expected Improvement, Thompson Sampling, Probability of Improvement, Upper Confidence Bound, etc.

$$y_{t+1} = f(x_{t+1}) \tag{4}$$

In some cases, a Gaussian noise term $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ is added to function evaluation as $f(x) + \varepsilon$ to make BO robust and not overfit.

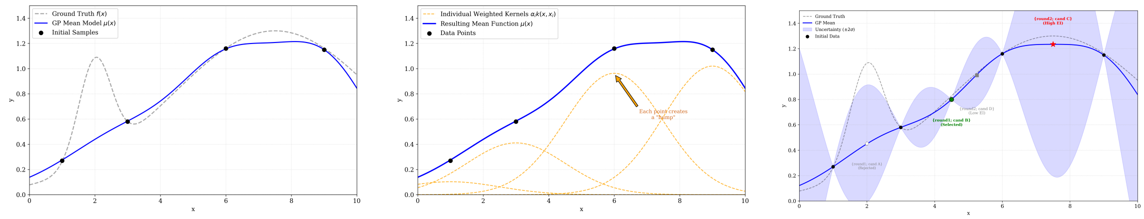
$$\mathcal{D}_{t+1} = \mathcal{D}_t \cup \{(x_{t+1}, y_{t+1})\} \tag{5}$$

Until a problem-specific goal or stopping-criteria is achieved, more samples are acquired and the *GP* is refit following Eqs. (2) - (5).

Applications of BO include hyper-parameter tuning in neural networks, kriging in mining, etc.

4.1.2 Attacks and Countermeasures.

High Resource Attack: A high resource attack also referred to as resource exhaustion attack is a type of attack in which an adversary aims to intentionally diminish the target's defined computational resources, including cpu cycles,memory allocation,bandwidth,or disk space leading the system or service unavailable to authentic users. Such attacks aim to overwhelm the target by provoking disproportionate demand for these resources,frequently recurring actions that exploit vulnerabilities in resource management.Salient characteristics of resource high resource attacks comprise their non-destructive nature to data integrity and confidentiality hence their primary goal is to disrupt the service availability ,degrade performance without altering or stealing information.



(a) initial GP fit vs actual objective function (b) surrogate model fit to the example dataset (how each sample contributes) (c) candidate selection rounds based on acquisition function validation

Fig. 2. AI summary (to add acquisition functions used, their equations from google docs, explanation of intial dataset used): Bayesian optimization workflow on an example dataset using a Gaussian Process surrogate. Starting from an initial set of sampled inputs, the surrogate model captures predictive mean and uncertainty over the objective function (Figure 2a). As observations from the example dataset are incorporated, the model progressively approximates the underlying function (Figure 2b). An acquisition function is then evaluated over the input space to balance exploration and exploitation, and the maximizer of this acquisition determines the next sample to be evaluated, shown as the selected candidate in Figure 2c.

The core principle behind resource exhaustion attacks resource asymmetry leads attackers to incur minimal cost while forcing the target system to perform disproportionately expensive operations.This imbalance gives adversaries an advantage to exploit and induce significant resource consumption using relatively low-rate or low-volume inputs.The on-demand resource allocation nature of modern systems particularly makes them vulnerable to resource exhaustion attack. Dynamically provisioned resources, such as buffers, threads, and connections, can accumulate when malicious requests exceed reclamation capacity, leading to a denial of service. Systems that lack adequate validation, rate limiting, or allocation controls are especially susceptible,enabling rapid exhaustion even at moderate attack rates.

These attacks are aggravated by persistence and amplification mechanisms. Persistence prolongs resource occupancy, while amplification enables low-cost inputs to trigger disproportionately expensive system responses.

Depending on the targeted resource, resource exhaustion attacks can be broadly classified as CPU exhaustion, memory exhaustion, bandwidth exhaustion, connection exhaustion, and disk exhaustion attacks.

Bayesian Optimization as High Resource Attack: Bayesian optimization has become a prominent technique in adversarial machine learning, particularly when dealing with the more realistic black-box setting that requires an attacker to find an adversarial perturbation without any knowledge of the architecture, parameters, or training data of the target model. In such cases, information about the model can only be obtained through queries, i.e. supplying an input to the model and receiving the corresponding output. In these situations, an attacker can treat interactions with the target model as a sequential optimization problem, in which like the model's response latency, predicted labels,confidence scores and resource utilization , serve as noisy measurements of some unknown underlying goal the attacker is trying to achieve.

When applied to high-resource attacks, BO enables an adversary to efficiently identify inputs that result in maximal computational cost during model inference. By approximating the behavior of the target model with a probabilistic surrogate and using acquisition functions to guide the selection of the query, Bayesian optimization allows an attacker to progressively identify inputs that induce execution paths of the worst-case.

Triggering such inputs in a loop may result in extensive resource exhaustion, leading to system degradation rendering the failure of service or denial of service. This trait of BO outperforms random search and heuristic approaches and makes it particularly effective for these attacks by achieving greater impact in fewer iterations. Despite their effectiveness, these attacks are constrained by the presence of resource control measures, including rate limiting, timeouts, and input validation. Furthermore, targeting well-provisioned systems typically requires distributed attack infrastructures, increasing the risk of detection. Hence, some mitigation strategies like limiting resource quotas,load balancing,anomaly detection and adaptive throttling, are critical to minimizing system vulnerability.

Countermeasures: Despite the fact that their effectiveness, these attacks are constrained by the presence of resource control measures, including rate limiting, timeouts, and input validation. Moreover, targeting well-resourced systems usually requires distributed attack infrastructures thereby increasing the risk of detection.

Responsibility: Amit Singh

Fill Tables 1 and 2.

Table 1. Comparison of Probing-Based Analysis Techniques

Paper (Domain)	Method	Conditions	Resources	System Knowledge	Countermeasures?	Verification Method	Limitations	Relevance	Notes
Dummy row - remove this[3](NFs)	Packets sent	Network jitter	High CPU, moderate memory	Black-box	Yes	Synthetic datasets	Limited scalability	High	Seminal

Table 2. Comparison of Countermeasures

Paper (Domain)	Method	Performance Impact	Resource Overhead	Test Method	Limitations	Relevance	Remarks
Dummy row - remove this[2](NFs)	BO	Latency increase under peak load	Moderate CPU and memory overhead	Real network	Reduced effectiveness under adaptive attacks	High	Dataset not shared

Write a few paragraphs on attacks and countermeasures here summarising each paper along the columns of Tables 1 and 2. Add relevant information, if any, not mentioned in the table. For example, how the probing and later the attack happens, whether there are trade-offs, etc. State the dominant features across the papers that you have studied that you want to highlight.

While you do this, if there is something that you find applicable to the section on open challenges and future research directions, add that there.

4.2 Low Rate Flow-table overflow attacks (Low Resource)

Responsibility: Robin

Copy Tables.1 and 2 here.

Write a few paragraphs on attacks and countermeasures here summarising each paper along the columns of Tables 1 and 2. Add relevant information, if any, not mentioned in the table. For example, how the probing and later the attack happens, whether there are trade-offs, etc. State the dominant features across the papers that you have studied that you want to highlight.

While you do this, if there is something that you find applicable to the section on open challenges and future research directions, add that there.

5 Open Challenges and Future Research Directions

5.1 Bayesian Optimization

5.1.1 Open Challenges.

5.1.2 Future Directions.

5.2 Low Rate Flow-table overflow attacks

5.2.1 Open Challenges.

5.2.2 Future Directions.

5.3

6 Conclusions

References

[1] Yehuda Afek, Harel Berger, and Anat Bremler-Barr. 2025. POPS: From History to Mitigation of DNS Cache Poisoning Attacks. *arXiv preprint arXiv:2501.13540* (2025).

[2] Abdussalam Ahmed Alashhab, Mohd Soperi Mohd Zahid, Mohamed A Azim, Muhammad Yunis Daha, Babangida Isyaku, and Shimhaz Ali. 2022. A survey of low rate DDoS detection techniques based on machine learning in software-defined networks. *Symmetry* 14, 8 (2022), 1563.

[3] Nirav Atre, Hugo Sadok, Erica Chiang, Weina Wang, and Justine Sherry. 2022. Surgeprotector: Mitigating temporal algorithmic complexity attacks using adversarial scheduling. In *Proceedings of the ACM SIGCOMM 2022 Conference*. 723–738.

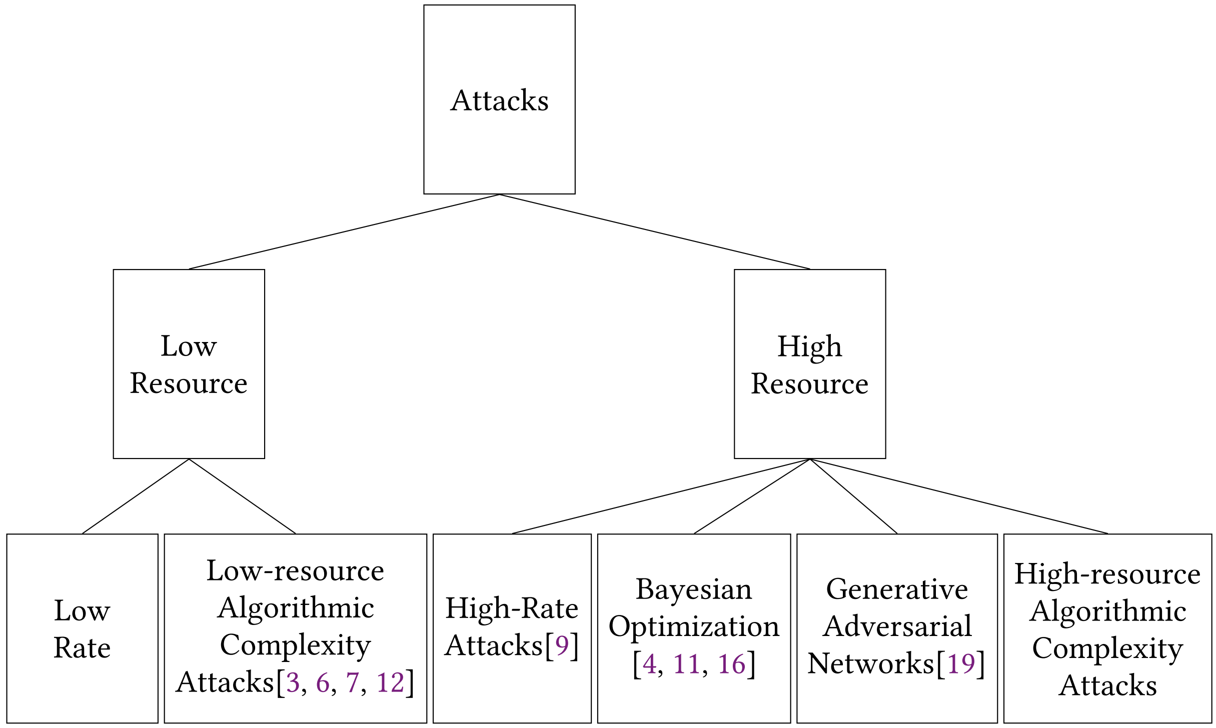


Fig. 3. Attack taxonomy

[4] Satyesh Shanker Awasthi, Mohammed Irshadh Ismaaeel Sathyamangalam Imran, Stefano Arrigoni, and Francesco Braghin. 2025. Bayesian Optimization applied for accelerated Virtual Validation of the Autonomous Driving Function. *arXiv preprint arXiv:2507.22769* (2025).

[5] Eric Brochu, Vlad M. Cora, and Nando De Freitas. 2010. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599* (2010). https://arxiv.org/abs/1012.2599

[6] Scott A Crosby and Dan S Wallach. 2003. Denial of service via algorithmic complexity attacks. In *12th USENIX Security Symposium (USENIX Security 03)*.

[7] Levente Csikor, Dinil Mon Divakaran, Min Suk Kang, Attila Kőrösi, Balázs Sonkoly, Dávid Haja, Dimitrios P Pezaros, Stefan Schmid, and Gábor Rétvári. 2019. Tuple space explosion: A denial-of-service attack against a software packet classifier. In *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*. 292–304.

[8] Milad Ghaznavi, Elaheh Jalalpour, Mohammad A Salahuddin, Raouf Boutaba, Daniel Migault, and Stere Preda. 2021. Content delivery network security: A survey. *IEEE Communications Surveys & Tutorials* 23, 4 (2021), 2166–2190.

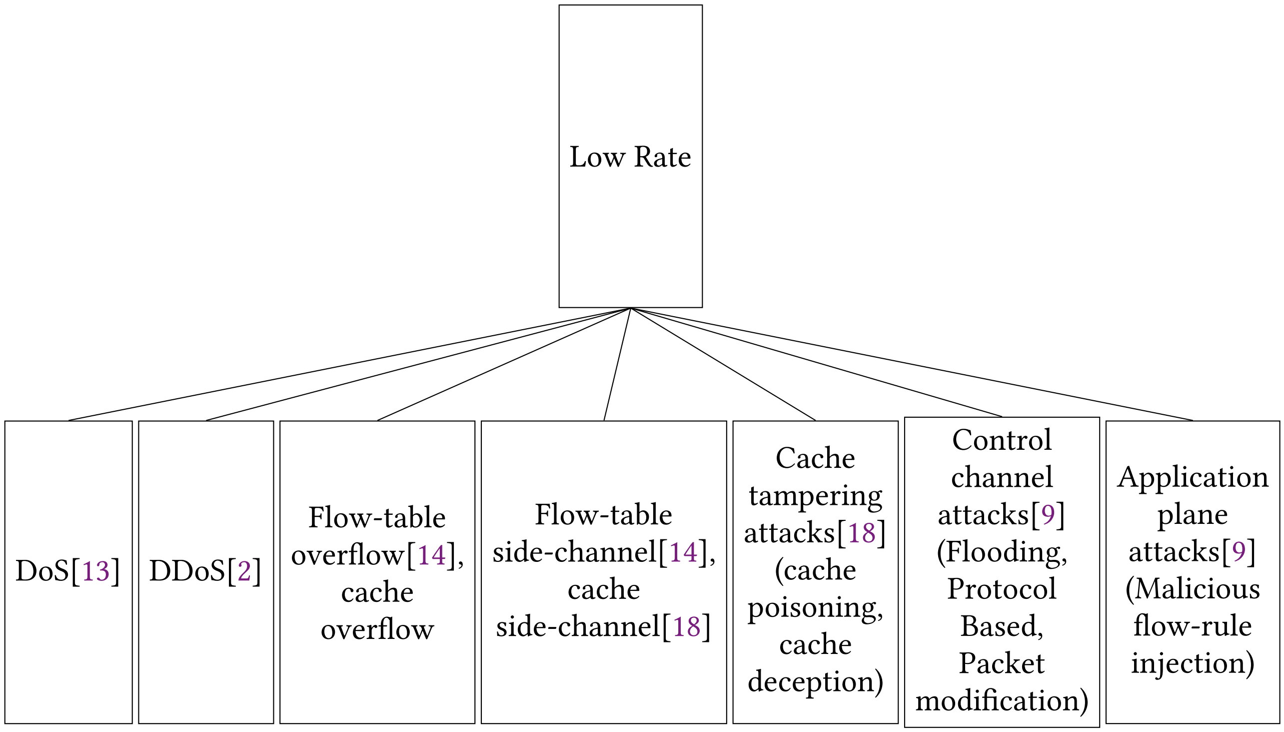


Fig. 4. Taxonomy of low-rate attacks

[9] Suruchi Karnani, Neha Agrawal, and Rohit Kumar. 2024. A comprehensive survey on low-rate and high-rate DDoS defense approaches in SDN: taxonomy, research challenges, and opportunities. *Multimedia Tools and applications* 83, 12 (2024), 35253–35306.

[10] Yangdi Lyu and Prabhat Mishra. 2018. A survey of side-channel attacks on caches and countermeasures. *Journal of Hardware and Systems Security* 2, 1 (2018), 33–50.

[11] Augusto Mondelli, Yueshan Li, Alessandro Zanardi, and Emilio Frazzoli. 2025. Test Automation for Interactive Scenarios via Promptable Traffic Simulation. *arXiv preprint arXiv:2506.01199* (2025).

[12] Theofilos Petsios, Jason Zhao, Angelos D Keromytis, and Suman Jana. 2017. Slowfuzz: Automated domain-independent detection of algorithmic complexity vulnerabilities. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*. 2155–2168.

[13] Vinícius De Miranda Rios, Pedro RM Inácio, Damien Magoni, and Mário M Freire. 2022. Detection and mitigation of low-rate denial-of-service attacks: A survey. *IEEE Access* 10 (2022), 76648–76668.

[14] Dan Tang, Rui Dai, Yudong Yan, Keqin Li, Wei Liang, and Zheng Qin. 2024. When sdn meets low-rate threats: A survey of attacks and countermeasures in programmable networks. *Comput. Surveys* 57, 4 (2024), 1–32.

[15] Tian Xie. 2024. *Resilient Cache Network Management: Algorithms, Analysis, Experiments.* The Pennsylvania State University.

[16] Johannes Zerwas, Patrick Kalmbach, Laurenz Henkel, Gábor Rétvári, Wolfgang Kellerer, Andreas Blenk, and Stefan Schmid. 2019. Netboa: Self-driving network benchmarking. In *Proceedings of the 2019 Workshop on Network Meets AI & ML*. 8–14.

[17] Jiliang Zhang, Congcong Chen, Jinhua Cui, and Keqin Li. 2024. Timing side-channel attacks and countermeasures in CPU microarchitectures. *Comput. Surveys* 56, 7 (2024), 1–40.

[18] Xianzhi Zhang, Yipeng Zhou, Di Wu, Quan Z. Sheng, Shazia Riaz, Miao Hu, and Linchang Xiao. 2025. A Survey on Privacy-Preserving Caching at Network Edge: Classification, Solutions, and Challenges. *Comput. Surveys* 57, 5 (2025), 1–38. doi:10.1145/3706630

[19] Yiran Zhu, Lei Cui, Zhenquan Ding, Lun Li, Yongji Liu, and Zhiyu Hao. 2022. Black box attack and network intrusion detection using machine learning for malicious traffic. *Computers & Security* 123 (2022), 102922.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009