

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, 19 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:

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.

Manuscript submitted to ACM

Manuscript submitted to ACM

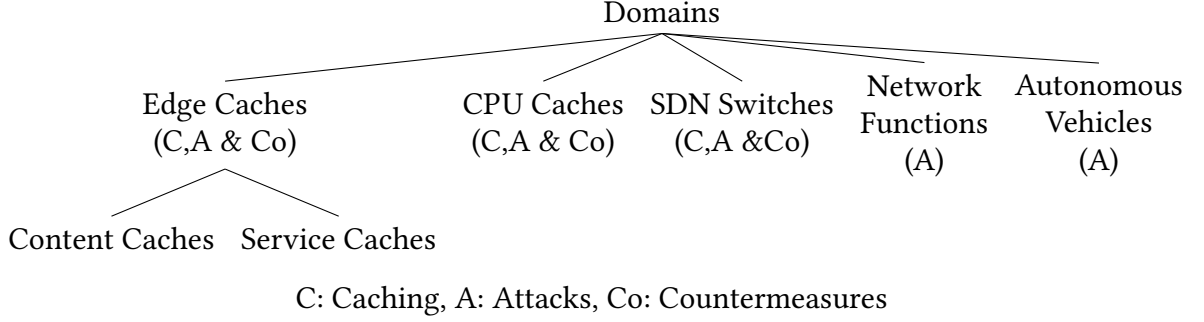


Fig. 1. Domains surveyed

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 further 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: \[10\], \[18\]](#)

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 [16] 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 Prevention 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 (heuristic, 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.).

Software Defined Networking (SDN) introduces a paradigm shift in network design by decoupling the control plane from the data plane. In traditional networks, forwarding devices such as switches and routers independently make packet forwarding decisions based on locally stored routing and forwarding logic. This tightly coupled architecture makes network management complex and inflexible.

In SDN, the control plane is logically centralized in an SDN controller, while the data plane consists of multiple SDN switches that perform packet forwarding. The controller acts as the network's decision-making entity, maintaining a global view of the network and dynamically installing forwarding rules on switches using standardized southbound interfaces such as OpenFlow.

SDN switches maintain flow tables that store flow rules defining how packets should be handled. These rules typically match on packet header fields such as source and destination IP addresses, transport-layer ports, and protocol identifiers. For performance reasons, flow rules are commonly stored in Ternary Content Addressable Memory (TCAM), which enables fast parallel lookups. However, TCAM is expensive and has limited capacity, making flow table space a scarce resource.

When a packet arrives at a switch, the switch performs a lookup in its flow table. If a matching rule is found, the packet is forwarded accordingly at line rate. If no matching rule exists, the packet (or its header) is forwarded to the controller via a Packet-In message. The controller then determines the appropriate forwarding action and may install a new flow rule in the switch. This behavior effectively treats the flow table as a cache, where cache hits result in fast forwarding and cache misses trigger interaction with the controller.

Due to the limited capacity of TCAM and the reactive nature of rule installation, SDN switches are vulnerable to attacks that aim to exhaust flow table space or overload the control channel. These characteristics make SDN switches a relevant domain for studying caching-related attacks and countermeasures.

2.2 Network Functions

Responsibility: Robin What are Network Functions?

Network Functions (NFs) are specialized components in modern networks that perform packet processing tasks beyond simple forwarding. Common examples include firewalls, intrusion detection and prevention systems (IDS/IPS), deep packet inspection (DPI) engines, load balancers, and network address translation (NAT) devices. These functions operate on traffic streams to enforce security policies, monitor behavior, transform packets, or extract application-level semantics.

Unlike traditional forwarding elements, NFs are often stateful and computation-intensive. Packet processing in an NF may involve maintaining per-flow state, traversing complex data structures, performing pattern matching, or executing protocol-specific logic. As a result, the amount of computation required to process a packet is frequently input-dependent and can vary significantly across packets and flows.

Modern deployments increasingly place NFs at the network edge to reduce latency and improve responsiveness for end users. However, edge environments typically operate under strict resource constraints, including limited CPU capacity, memory, and bandwidth. To achieve high performance under these constraints, NFs rely on internal data structures and fast-path mechanisms that resemble caching systems, where frequently accessed state or computation results are stored in limited, high-speed resources.

These characteristics make NFs an important domain for studying attacks and countermeasures related to resource exhaustion and performance degradation. In particular, the combination of shared resources, stateful processing, and input-dependent computation exposes NFs to adversarial behaviors that can disproportionately impact system performance, even when the attacker's resource investment is modest.

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 a powerful model-based sequential strategy to efficiently identify the global optimum of an objective function and the optimizer in the input domain. Generally, an optimization problem is also formulated as maximizing the function $f(x)$ in a domain $\mathcal{X} \subset \mathbb{R}^d$, i.e., $\max_{x \in \mathcal{X} \subset \mathbb{R}^d} f(x)$. BO is appropriate for functions with a few specific characteristics 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), sequential decision-making (acquisition functions such as expected improvement), and surrogate modeling (Gaussian Process).

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

$$p(M | D) = \frac{p(D | M) p(M)}{p(D)}$$

since $p(D)$ does not depend on M , it is also commonly represented as $p(M | D) \propto p(D | M) p(M)$ in the BO literature. Along the lines of Bayes' theorem, BO assumes a prior belief for the unknown objective function $f(x)$ called the prior probability distribution over $f(x)$ (i.e., $f(x) \sim \text{prior}$), uses the initial samples in the dataset D to update the *prior* to the *posterior* probability distribution. To efficiently acquire more data points for D , BO

uses an acquisition function that guides the search and validates candidates in the search space of x . One of the candidates is acquired as a new data point, then evaluated for y using $f(x)$, and is used to update *posterior*. BO continues to acquire more data until a stopping criterion or maximum iterations is met. A good BO run should identify a global optimum (peak of $(f(x))$ and optimizer (corresponding x) in fewer iterations.

The $f(x)$ is modeled as a function sampled from a probability distribution over functions. The prior belief in BO represents the space of all possible objective functions (function space \mathcal{H} where $f(x)$ lies) and is an inductive bias that encodes assumptions about the nature of $f(x)$, such as smoothness, continuity, differentiability, noise, etc., that make some possible functions more plausible. The prior belief includes the choice of statistical process and its hyper-parameters, 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, which collectively describe the nature of the function $f(x)$. For this reason, the prior affects the sampling efficiency in acquisition functions, overall BO convergence, and thus the choice of prior is crucial.

Some works assume a Lipschitz-continuous or lives in a Reproducing Kernel Hilbert Space (RKHS) objective function to prove theoretical guarantees of BO convergence.?

BO conditions on a dataset D with initial samples, where D is a set of 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 $f(x)$ evaluate x to y . A prior belief, also referred to as a surrogate model, is assumed on the basis of characteristics of the objective function. In BO literature, the standard surrogate model for $f(x)$ is a Gaussian Process (GP). GPs are a rich and flexible class of non-parametric statistical models over function spaces with domains that can be continuous, discrete, hybrid, or hierarchical. A GP is intuitively understood as a distribution over functions; each sample drawn from a GP corresponds to a mathematical function. A mean function and a kernel function (also referred to as the pair-wise covariance function of the samples in D) parameterize the characteristics of this distribution. The Eq. below represents the GP. If $x_t \in \mathbb{R}^d$, the GP is d -dimensional, and a 0-dimensional GP can be understood as a Normal distribution. See an example BO run in figure 2. Examples of other surrogate models include the Student-t Process, etc.

$$f(x) \sim \mathcal{GP}(m(x), k(x, x'))$$

where $m(x)$ represents the mean function and $k(x, x')$ represents the kernel function of GP. Applying Bayes' theorem to GP leads to the inference rules 2.1, 2.2 for the posterior mean function ($\mu_t(x)$) and the posterior variance function (σ_t^2) after the t samples are appended to the initial i in D . After each refit, the posterior probability mass increasingly concentrates around the true objective function $f(x)$, decreasing the uncertainty in the regions where samples were drawn, as shown in 1. Generally, $m(x) = 0$ and $k(x, x) = 1$. A few choices for kernel function include Squared Exponential (RBF) kernel, Matérn kernel, Rational Quadratic (RQ) kernel, etc.

$$f(x) \sim p(f(x) \mid \mathcal{D}_t) \quad (1)$$

The kernel matrix K , calculated using D , represents pair-wise covariance. The *posterior* is fit using the initial i samples in D (the i is a hyper-parameter and is problem-specific). Some candidates, in the domain of $f(x)$, are collected, validated by the acquisition function to select the next-best sample, which is evaluated using the actual objective function $f(x)$, and finally added to D . This acquired sample is used to refit *posterior*, see 2.1, 2.2 and 3 - 5.

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

The $\mathbf{k}_*(x_t) \in \mathbb{R}^{i+t-1}$ in 2.1 represents the covariance vector of x_t with the other $i + t - 1$ samples in D , $\mathbf{K}_{t-1} \in \mathbb{R}^{i+t-1 \times i+t-1}$ represents the pair-wise covariance matrix of $i + t - 1$ samples, and $\mathbf{y}_{1:i+t-1} \in \mathbb{R}^{i+t-1}$ represents the vector of values y of all $i + t - 1$ samples in D . Intuitively, the pairwise covariance of the $x_{1:i+t-1}$ data points is projected onto the y space and weighted by the covariance vector of the new point x_t .

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

Similarly to 2.1, the second-term represents the information gain based on covariance values subtracted from variance $k(x, x)$.

Generally, to acquire a new sample, candidates are sampled throughout the search space of x . Using Eqs. 2.1, 2.1, the mean function and the kernel function are inferred, based on *posterior*, for a candidate x_{t_c} of the acquisition function α . A gradient-based optimization technique, such as the L-BFGS-B algorithm, is used to estimate the best value

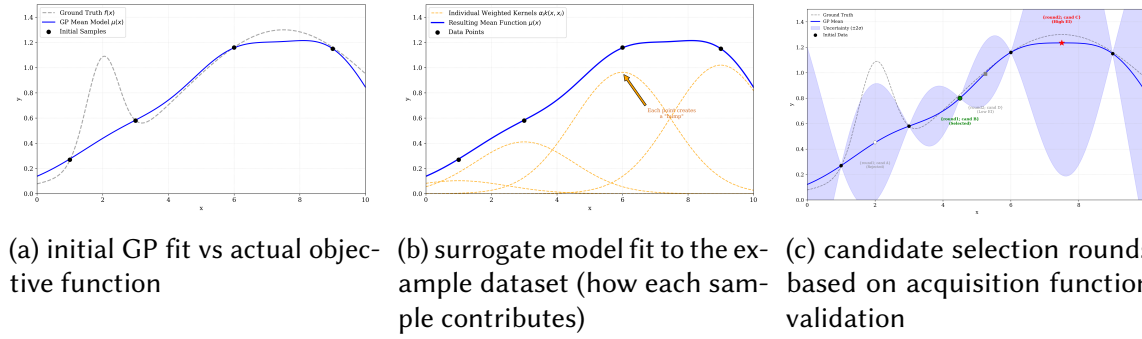


Fig. 2. The figure shows an example BO run. A true objective function is supposed as $f(x) = 0.8 \exp\left(-\frac{(x-2)^2}{0.5}\right) + 1.3 \exp\left(-\frac{(x-7.5)^2}{20}\right)$. The initial dataset points are $X = [1, 3, 6, 9]^T$, $y = [0.27, 0.58, 1.16, 1.15]^T$. The kernel function squared exponential $k(x, x') = \exp\left(-\frac{(x-x')^2}{4.5}\right)$ and acquisition function expected improvement $EI(x) = (\mu(x) - f^+ - \xi)\Phi(Z) + \sigma(x)\phi(Z)$ were used. Where f^+ is the current best observed value, $\xi = 0.01$, and $Z = \frac{\mu(x) - f^+ - \xi}{\sigma(x)}$. [Gem]

of y corresponding to x values in the candidate space as x_t .

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

Eq. 3 is the acquisition function α to efficiently select the next sample x_t to evaluate the true objective function $f(x)$ 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.

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

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

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

Until a problem-specific goal or stopping criterion is achieved, more samples are acquired, and GP is refitted following Eqs. 3 - 5.

BO is applicable wherever there is a black-box entity whose properties are to be efficiently derived. BO has diverse applications that span hyper-parameter tuning in neural networks, kriging in mining, novel drug trials, and self-benchmarking models. There are

a number of Python/C packages for a faster and robust implementation of BO, such as *skopt*, *BOTorch*, and *BayesOpt*.

In the surveyed literature, Bayesian Optimization consistently exhibits three dominant properties: (i) query-efficiency in high-dimensional black-box spaces, (ii) robustness to noise through probabilistic modeling, and (iii) the ability to explicitly model uncertainty through posterior variance. These properties make BO particularly suitable for both adversarial workload discovery and defensive hardening in resource-constrained edge caching systems.

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.

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.

Table 1. Comparison of Probing-Based Analysis Techniques

<i>Paper (Domain)</i>	<i>Method</i>	<i>Conditions</i>	<i>Resources</i>	<i>System Knowledge</i>	<i>Countermeasures?</i>	<i>Verification Method</i>	<i>Limitations</i>	<i>Relevance</i>	<i>Notes</i>
Dummy row remove this[3](NFs)	Packets - sent	Network jitter	High CPU, mod- erate memory	Black- box	Yes	Synthetic datasets	Limited scalabil- ity	High	Seminal

Responsibility: Amit Singh

Fill Tables 1 and 2.

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.

NetBOA: Self-Driving Network Benchmarking Zerwas et al. [17] propose NetBOA, a framework for automating the generation of adversarial network traffic workloads to benchmark black-box network functions. The authors formulate the discovery of performance bottlenecks—such as high CPU utilization or latency—as a Bayesian Optimization problem. NetBOA models the unknown relationship between traffic configuration parameters (e.g., inter-arrival times, burst sizes) and the target performance metric using a Gaussian Process (GP) prior with a Matérn kernel. To navigate the configuration space \mathcal{X} , the framework employs the Expected Improvement (EI) acquisition function, effectively balancing exploration of unseen configurations and exploitation of known high-cost regions. The approach demonstrates that "self-driving" benchmarks can identify worst-case

Manuscript submitted to ACM

Table 2. Comparison of Countermeasures

<i>Paper (Domain)</i>	<i>Method</i>	<i>Performance Impact</i>	<i>Resource Overhead</i>	<i>Test Method</i>	<i>Limitations</i>	<i>Relevance</i>	<i>Remarks</i>
Dummy row remove this[2](NFs)	BO -	Latency increase under peak load	Moderate CPU and memory overhead	Real net-work	Reduced effectiveness under adaptive attacks	High	Dataset not shared
Wang et al. [15] (PHEV EMS)	BO (Matlab default RBF + EI)	Improved fuel economy, convergence stability, and robustness of EMS	Offline tuning cost using BO (no runtime overhead)	Simulation (MATLAB/Simulink vehicle model)	No real-world deployment	Medium	Focuses on hyperparameter tuning, not adversarial robustness
NetBOA [17] (Network / OVS Cache)	BO (Matern 5/2? + EI)	Forces worst-case cache execution paths; increases lookup latency and CPU utilization - ?	High query budget; moderate-to-high CPU stress on target during attack - ?	Black-box query-based evaluation on Open vSwitch (OVS) cache - ?	Effectiveness depends on query access and noise stability; may be mitigated by rate limiting or adaptive caching - ?	High	Formulates attack as sequential optimization; to identify inputs maximizing computational cost. - ?

execution paths in systems like Open vSwitch (OvS) significantly faster than random search, highlighting the utility of BO in uncovering algorithmic complexity attacks.

Auto-Tuning Vehicle Dynamics (APOVD) Wang et al. [?] introduce the Automatic Parameter Optimization of Vehicle Dynamics (APOVD) framework to bridge the reality gap between high-fidelity physical vehicle models and real-world 4WID electric vehicles. The authors treat the calibration of the 8-DOF vehicle dynamics model as a black-box optimization problem, aiming to minimize the modeling error $J = ||y_{real} - f_{ODE}(\theta)||$. The study compares Single-Objective BO (using GP priors with EI or UCB acquisition functions) against Multi-Objective BO. Notably, the Multi-Objective approach utilizing a Probabilistic Random Forest (PRF) surrogate and Expected Hypervolume Improvement (EHVI) acquisition function achieved the highest accuracy, reducing the Root Mean Square Error (RMSE) of longitudinal velocity and yaw rate by over 90

Bayesian optimization is used for hyperparameter tuning in TD3-based EMS systems [15].

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.

Manuscript submitted to ACM

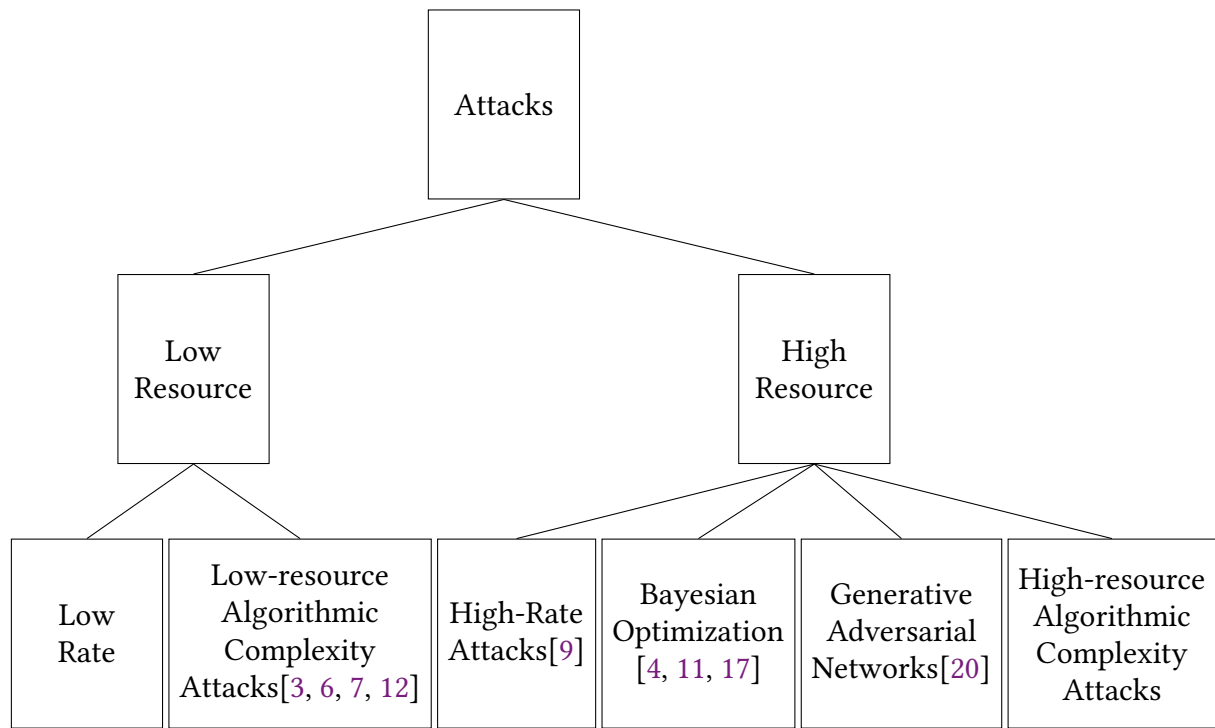


Fig. 3. Attack taxonomy

5.2.2 Future Directions.

5.3

6 Miscellaneous

Add here :papers that you think are interesting but do not fit into the above categories.

7 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 Doha, 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.

Manuscript submitted to ACM

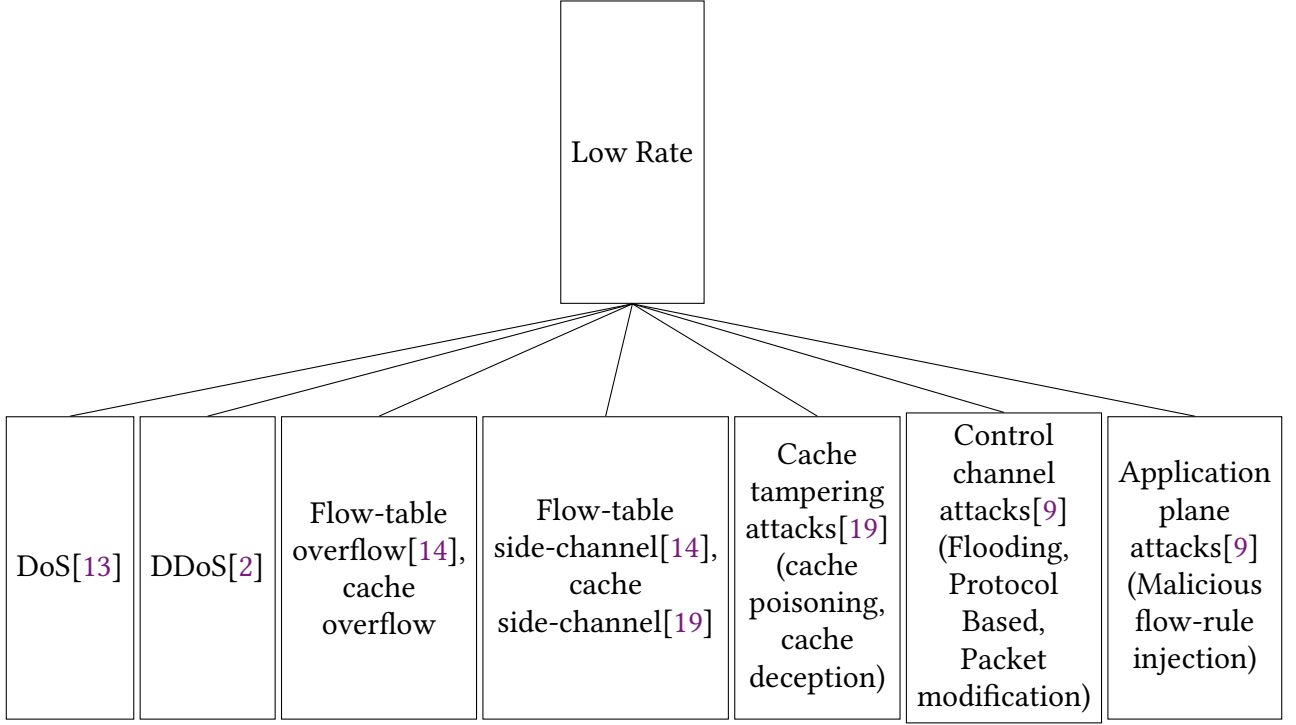


Fig. 4. Taxonomy of low-rate attacks

- [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.
- [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.
- [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] Jinhai Wang, Changqing Du, Fuwu Yan, Min Hua, Xiangyu Gongye, Quan Yuan, Hongming Xu, and Quan Zhou. 2025. Bayesian optimization for hyper-parameter tuning of an improved twin delayed deep deterministic policy gradients based energy management strategy for plug-in hybrid electric vehicles. *Applied Energy* 381 (2025), 125171. doi:10.1016/j.apenergy.2024.125171
- [16] Tian Xie. 2024. *Resilient Cache Network Management: Algorithms, Analysis, Experiments*. The Pennsylvania State University.
- [17] 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.
- [18] 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.
- [19] 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
- [20] 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

Manuscript submitted to ACM