

1           **Caching systems: attacks and**  
2  
3           **countermeasures - A Survey**  
4  
5

6           Write the abstract here.  
7

8           CCS Concepts: • **Do Not Use This Code → Generate the Correct Terms for Your Paper;**  
9           *Generate the Correct Terms for Your Paper; Generate the Correct Terms for Your Paper; Generate*  
10          *the Correct Terms for Your Paper.*  
11

12          Additional Key Words and Phrases: Do, Not, Use, This, Code, Put, the, Correct, Terms, for, Your,  
13          Paper  
14

15          **ACM Reference Format:**

16          . 2025. Caching systems: attacks and countermeasures - A Survey. In *Proceedings of Make sure*  
17          *to enter the correct conference title from your rights confirmation email (Conference acronym 'XX).*  
18          ACM, New York, NY, USA, 19 pages. <https://doi.org/XXXXXXX.XXXXXXX>  
19

20  
21          **1 Introduction**

22          **Responsibility:** Radhika  
23

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

---

28          Author's Contact Information:  
29

30          Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted  
31          without fee provided that copies are not made or distributed for profit or commercial advantage and that  
32          copies bear this notice and the full citation on the first page. Copyrights for components of this work owned  
33          by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise,  
34          or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.  
35          Request permissions from permissions@acm.org.  
36

37          © 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.  
38

39          Manuscript submitted to ACM  
40

41          Manuscript submitted to ACM

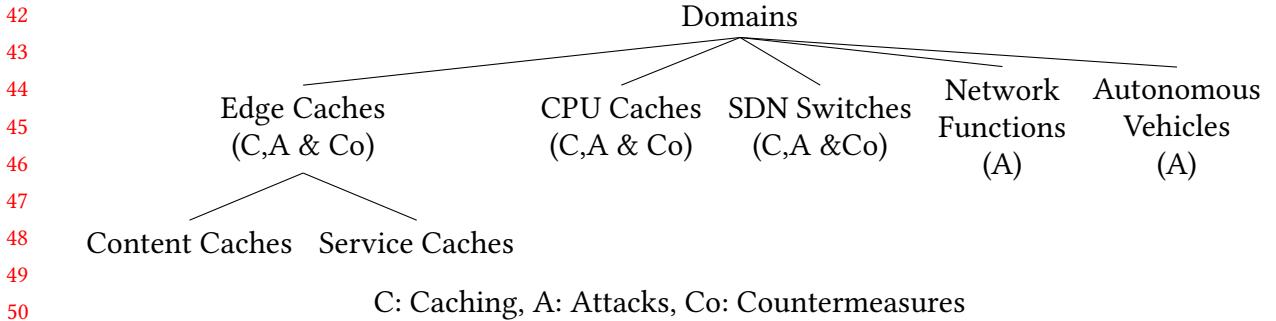


Fig. 1. Domains surveyed

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 Accessible 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,

124 connectivity, and traffic locality, while content placement, DNS-based request steering,  
125 and caching decisions are centrally managed by the CDN. CDNs cache objects and small  
126 state-less functions, unlike an edge cache, which can cache services too. [Read \[8\] for more](#)  
127 [on this](#).

128

129 Note: Caching in the World Wide Web and Information Centric Networking may also  
130 be studied. See [16] and associated papers.

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

137

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

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

151 We organize prior work along the following axes: attack methods and countermeasures,  
152 and domains surveyed. We examine attack methods according to their:  
153

- 154
- 155 • method of probing (reconnaissance), if applicable or if any and what is probed  
156 (replacement algorithm, size of cache, etc.)
- 157 • conditions for probing (do they consider network jitter, link delays, link failures,  
158 etc.)

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

173  
174 We examine countermeasures according to their:

- 175 • method (heuristic, AI etc.)
- 176 • impact on performance
- 177 • impact on resource consumption,
- 178 • tested on real networks or simulators, the datasets used, if any
- 179 • relevance to edge service caching
- 180 • limitations (are there strong assumptions, are the datasets realistic, etc.)

### 186 1.1 Comparison with existing surveys

187 Responsibility: TBD

### 190 1.2 Why this survey is different

191 Responsibility: TBD

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

194 Our contributions:

## 198 2 Background and system model

199 Responsibility: TBD

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

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

**206 2.1 SDN Switches**

207 Responsibility: Robin.

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

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

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

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

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

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

**247 2.2 Network Functions****248 Responsibility: Robin What are Network Functions?**

249  
250 Network Functions (NFs) are specialized components in modern networks that perform  
251 packet processing tasks beyond simple forwarding. Common examples include firewalls,  
252 intrusion detection and prevention systems (IDS/IPS), deep packet inspection (DPI) en-  
253 gines, load balancers, and network address translation (NAT) devices. These functions  
254 operate on traffic streams to enforce security policies, monitor behavior, transform packets,  
255 or extract application-level semantics.

256  
257 Unlike traditional forwarding elements, NFs are often stateful and computation-intensive.  
258 Packet processing in an NF may involve maintaining per-flow state, traversing complex  
259 data structures, performing pattern matching, or executing protocol-specific logic. As  
260 a result, the amount of computation required to process a packet is frequently input-  
261 dependent and can vary significantly across packets and flows.

262  
263 Modern deployments increasingly place NFs at the network edge to reduce latency and  
264 improve responsiveness for end users. However, edge environments typically operate  
265 under strict resource constraints, including limited CPU capacity, memory, and bandwidth.  
266 To achieve high performance under these constraints, NFs rely on internal data structures  
267 and fast-path mechanisms that resemble caching systems, where frequently accessed state  
268 or computation results are stored in limited, high-speed resources.

269  
270 These characteristics make NFs an important domain for studying attacks and coun-  
271 termeasures related to resource exhaustion and performance degradation. In particular,  
272 the combination of shared resources, stateful processing, and input-dependent compu-  
273 tation exposes NFs to adversarial behaviors that can disproportionately impact system  
274 performance, even when the attacker's resource investment is modest.

**275 2.3 Relevance to edge service caching**

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

288 **3 Attack taxonomy**

289 Refer Fig.3 and 4 for the attack taxonomy.

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

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

293  
294 **4 Attacks and countermeasures**

295 **4.1 Bayesian Optimization (High resource attacks)**

296     **4.1.1 Introduction to Bayesian Optimization.** Responsibility: Vasudeva

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

299     Bayesian Optimization (BO) is a powerful model-based sequential strategy to efficiently  
300     identify the global optimum of an objective function and the optimizer in the input domain.  
301     Generally, an optimization problem is also formulated as maximizing the function  $f(x)$  in  
302     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  
303     characteristics such as costly to evaluate, unknown/no mathematical representation, no  
304     access to derivatives, non-convex, limited observations, noisy observations, etc. [5].

305     BO is a strategy combining ideas from Bayesian inference (Bayes' theorem), sequential  
306     decision-making (acquisition functions such as expected improvement), and surrogate  
307     modeling (Gaussian Process).

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

$$312 \quad p(M | D) = \frac{p(D | M) p(M)}{p(D)}$$

313     since  $p(D)$  does not depend on  $M$ , it is also commonly represented as  $p(M | D) \propto p(D |$   
314      $M) p(M)$  in the BO literature. Along the lines of Bayes' theorem, BO assumes a prior  
315     belief for the unknown objective function  $f(x)$  called the prior probability distribution  
316     over  $f(x)$  (i.e.,  $f(x) \sim \text{prior}$ ), uses the initial samples in the dataset  $D$  to update the *prior*  
317     to the *posterior* probability distribution. To efficiently acquire more data points for  $D$ , BO  
318

319     Manuscript submitted to ACM

320  
321  
322  
323  
324  
325  
326  
327  
328

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

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

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

350 BO conditions on a dataset  $D$  with initial samples, where  $D$  is a set of  $i$  pairs of  $x, y$  and  
 351  $x \in \mathbb{R}^d, y \in \mathbb{R}$ , i.e.,  $D = \{x_{1:i}, y_{1:i}\}$  and  $f(x)$  evaluate  $x$  to  $y$ . A prior belief, also referred to  
 352 as a surrogate model, is assumed on the basis of characteristics of the objective function.  
 353 In BO literature, the standard surrogate model for  $f(x)$  is a Gaussian Process (GP). GPs  
 354 are a rich and flexible class of non-parametric statistical models over function spaces  
 355 with domains that can be continuous, discrete, hybrid, or hierarchical. A GP is intuitively  
 356 understood as a distribution over functions; each sample drawn from a GP corresponds  
 357 to a mathematical function. A mean function and a kernel function (also referred to as  
 358 the pair-wise covariance function of the samples in  $D$ ) parameterize the characteristics of  
 359 this distribution. The Eq. below represents the GP. If  $x_t \in \mathbb{R}^d$ , the GP is d-dimensional,  
 360 and a 0-dimensional GP can be understood as a Normal distribution. See an example BO  
 361 run in figure 2. Examples of other surrogate models include the Student-t Process, etc.  
 362

$$365 \quad f(x) \sim \mathcal{GP}(m(x), k(x, x')) \\ 366$$

370 where  $m(x)$  represents the mean function and  $k(x, x')$  represents the kernel function of  
 371 GP. Applying Bayes' theorem to GP leads to the inference rules 2.1, 2.2 for the posterior  
 372 mean function ( $\mu_t(x)$ ) and the posterior variance function ( $\sigma_t^2$ ) after the  $t$  samples are  
 373 appended to the initial  $i$  in  $D$ . After each refit, the posterior probability mass increasingly  
 374 concentrates around the true objective function  $f(x)$ , decreasing the uncertainty in the  
 375 regions where samples were drawn, as shown in 1. Generally,  $m(x) = 0$  and  $k(x, x) = 1$ . A  
 376 few choices for kernel function include Squared Exponential (RBF) kernel, Matérn kernel,  
 377 Rational Quadratic (RQ) kernel, etc.  
 378

$$380 \quad f(x) \sim p(f(x) | \mathcal{D}_t) \quad (1)$$

381 The kernel matrix  $K$ , calculated using  $D$ , represents pair-wise covariance. The *posterior*  
 382 is fit using the initial  $i$  samples in  $D$  (the  $i$  is a hyper-parameter and is problem-specific).  
 383 Some candidates, in the domain of  $f(x)$ , are collected, validated by the acquisition function  
 384 to select the next-best sample, which is evaluated using the actual objective function  $f(x)$ ,  
 385 and finally added to  $D$ . This acquired sample is used to refit *posterior*, see 2.1, 2.2 and 3  
 386 - 5.

$$389 \quad \mu_t(x_{t_c}) = \mathbf{m}(\mathbf{x}) + \mathbf{k}_*(x_t)^\top (\mathbf{K}_{t-1})^{-1} \mathbf{y}_{1:i+t-1} \quad (2.1)$$

390 The  $k_*(x_t) \in \mathbb{R}^{i+t-1}$  in 2.1 represents the covariance vector of  $x_t$  with the other  $i + t - 1$   
 391 samples in  $D$ ,  $K_{t-1} \in \mathbb{R}^{i+t-1 \times i+t-1}$  represents the pair-wise covariance matrix of  $i + t - 1$   
 392 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  
 393  $D$ . Intuitively, the pairwise covariance of the  $x_{1:i+t-1}$  data points is projected onto the  $y$   
 394 space and weighted by the covariance vector of the new point  $x_t$ .  
 395

$$398 \quad \sigma_t^2(x_t) = k(x_t, x_t) - \mathbf{k}_*(x_t)^\top (\mathbf{K}_{t-1})^{-1} \mathbf{k}_*(x_t) \quad (2.2)$$

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

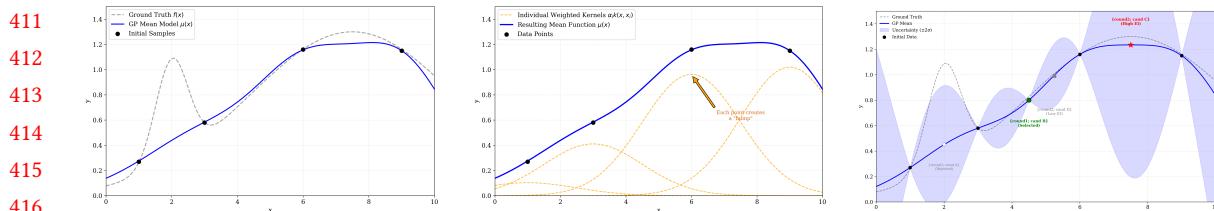
402 Generally, to acquire a new sample, candidates are sampled throughout the search  
 403 space of  $x$ . Using Eqs. 2.1, 2.1, the mean function and the kernel function are inferred,  
 404 based on *posterior*, for a candidate  $x_{t_c}$  of the acquisition function  $\alpha$ . A gradient-based  
 405 optimization technique, such as the L-BFGS-B algorithm, is used to estimate the best value  
 406

407

408

409 Manuscript submitted to ACM

410



(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. 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  $\alpha$  to efficiently select the next sample  $x_t$  to evaluate the true objective function  $f(x)$  for  $y_t$ . An ideal choice of  $\alpha$  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

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

*454* In the surveyed literature, Bayesian Optimization consistently exhibits three dominant  
*455* properties: (i) query-efficiency in high-dimensional black-box spaces, (ii) robustness to  
*456* noise through probabilistic modeling, and (iii) the ability to explicitly model uncertainty  
*457* through posterior variance. These properties make BO particularly suitable for both  
*458* adversarial workload discovery and defensive hardening in resource-constrained edge  
*459* caching systems.  
*460*

*461*

#### *462* 4.1.2 Attacks and Countermeasures.

*463*

*464* *High Resource Attack:* A high resource attack also referred to as resource exhaustion at-  
*465* tack is a type of attack in which an adversary aims to intentionally diminish the target’s de-  
*466* fined computational resources, including cpu cycles, memory allocation, bandwidth, or disk  
*467* space leading the system or service unavailable to authentic users. Such attacks aim to over-  
*468* whelm the target by provoking disproportionate demand for these resources, frequently  
*469* recurring actions that exploit vulnerabilities in resource management. Salient character-  
*470* istics of resource high resource attacks comprise their non-destructive nature to data  
*471* integrity and confidentiality hence their primary goal is to disrupt the service availability  
*472* , degrade performance without altering or stealing information.  
*473*

*474* The core principle behind resource exhaustion attacks resource asymmetry leads attack-  
*475* ers to incur minimal cost while forcing the target system to perform disproportionately  
*476* expensive operations. This imbalance gives adversaries an advantage to exploit and induce  
*477* significant resource consumption using relatively low-rate or low-volume inputs. The on-  
*478* demand resource allocation nature of modern systems particularly makes them vulnerable  
*479* to resource exhaustion attack. Dynamically provisioned resources, such as buffers, threads,  
*480* and connections, can accumulate when malicious requests exceed reclamation capacity,  
*481* leading to a denial of service. Systems that lack adequate validation, rate limiting, or  
*482* allocation controls are especially susceptible, enabling rapid exhaustion even at moderate  
*483* attack rates.  
*484*

*485* These attacks are aggravated by persistence and amplification mechanisms. Persistence  
*486* prolongs resource occupancy, while amplification enables low-cost inputs to trigger  
*487* disproportionately expensive system responses.  
*488*

*489* Manuscript submitted to ACM  
*490*

*491*

*492*

493     Depending on the targeted resource, resource exhaustion attacks can be broadly classi-  
494     fied as CPU exhaustion, memory exhaustion, bandwidth exhaustion, connection exhaus-  
495     tion, and disk exhaustion attacks.  
496

497     *Bayesian Optimization as High Resource Attack:* Bayesian optimization has become a  
498     prominent technique in adversarial machine learning, particularly when dealing with the  
499     more realistic black-box setting that requires an attacker to find an adversarial perturba-  
500     tion without any knowledge of the architecture, parameters, or training data of the target  
501     model. In such cases, information about the model can only be obtained through queries,  
502     i.e. supplying an input to the model and receiving the corresponding output. In these situ-  
503     ations, an attacker can treat interactions with the target model as a sequential optimization  
504     problem, in which like the model's response latency, predicted labels, confidence scores  
505     and resource utilization , serve as noisy measurements of some unknown underlying goal  
506     the attacker is trying to achieve.  
507

508     When applied to high-resource attacks, BO enables an adversary to efficiently identify  
509     inputs that result in maximal computational cost during model inference. By approximat-  
510     ing the behavior of the target model with a probabilistic surrogate and using acquisition  
511     functions to guide the selection of the query, Bayesian optimization allows an attacker to  
512     progressively identify inputs that induce execution paths of the worst-case.  
513

514     Triggering such inputs in a loop may result in extensive resource exhaustion, leading to  
515     system degradation rendering the failure of service or denial of service. This trait of BO  
516     outperforms random search and heuristic approaches and makes it particularly effective  
517     for these attacks by achieving greater impact in fewer iterations. Despite their effectiveness,  
518     these attacks are constrained by the presence of resource control measures, including rate  
519     limiting, timeouts, and input validation. Furthermore, targeting well-provisioned systems  
520     typically requires distributed attack infrastructures, increasing the risk of detection. Hence,  
521     some mitigation strategies like limiting resource quotas, load balancing, anomaly detection  
522     and adaptive throttling, are critical to minimizing system vulnerability.  
523

524     *Countermeasures:* Despite the fact that their effectiveness, these attacks are constrained  
525     by the presence of resource control measures, including rate limiting, timeouts, and input  
526     validation. Moreover, targeting well-resourced systems usually requires distributed attack  
527     infrastructures thereby increasing the risk of detection.  
528

Table 1. Comparison of Probing-Based Analysis Techniques

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

543                   544                   545                   546                   547  
 548                   549                   550                   551                   552  
 553                   554                   555                   556                   557  
 558                   559                   560                   561                   562  
 563                   564                   565                   566                   567  
 568                   569                   570                   571                   572  
 573                   574

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

Paper (Domain)	Method	Performance Impact	Resource Overhead	Test Method	Limitations	Relevance	Remarks		
575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615	Dummy row - remove this[2](NFs)	BO	Latency increase under peak load	Moderate CPU and memory overhead	Real work	net-work overhead	Reduced effectiveness under adaptive attacks	High	Dataset not shared
Wang et al. [15] (PHEV EMS)	BO (Matlab default RBF + EI)	Improved economy, convergence and robustness of EMS	Offline tuning cost using BO runtime overhead)	Simulation (MATLAB/Simulink)	No vehicle model	real-world deployment	Medium	Focuses on hyper-parameter tuning, not adversarial robustness	
NetBOA [17]	BO (Network Kernel Cache)	Forces worst-case / OVS Kernel 5/2? + EI	High query budget; execution paths; increases lookup latency and CPU utilization - ?	Black-box query-based moderate evaluation on Open vSwitch (OVS) cache - ?	Effectiveness depends on query and noise stability; may be mitigated by rate limiting or adaptive caching - ?	High access	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.

616 Auto-Tuning Vehicle Dynamics (APOVD) Wang et al. [? ] introduce the Automatic  
 617 Parameter Optimization of Vehicle Dynamics (APOVD) framework to bridge the reality  
 618 gap between high-fidelity physical vehicle models and real-world 4WID electric vehicles.  
 619 The authors treat the calibration of the 8-DOF vehicle dynamics model as a black-box  
 620 optimization problem, aiming to minimize the modeling error  $J = ||y_{real} - f_{ODE}(\theta)||$ .  
 621 The study compares Single-Objective BO (using GP priors with EI or UCB acquisition  
 622 functions) against Multi-Objective BO. Notably, the Multi-Objective approach utilizing a  
 623 Probabilistic Random Forest (PRF) surrogate and Expected Hypervolume Improvement  
 624 (EHVI) acquisition function achieved the highest accuracy, reducing the Root Mean Square  
 625 Error (RMSE) of longitudinal velocity and yaw rate by over 90  
 626

627 Bayesian optimization is used for hyperparameter tuning in TD3-based EMS systems  
 628 [15].  
 629

## 630 4.2 Low Rate Flow-table overflow attacks (Low Resource)

631 **Responsibility:** Robin  
 632

633 Copy Tables 1 and 2 here.  
 634

635 Write a few paragraphs on attacks and countermeasures here summarising each paper  
 636 along the columns of Tables 1 and 2. Add relevant information, if any, not mentioned in  
 637 the table. For example, how the probing and later the attack happens, whether there are  
 638 trade-offs, etc. State the dominant features across the papers that you have studied that  
 639 you want to highlight.  
 640

641 While you do this, if there is something that you find applicable to the section on open  
 642 challenges and future research directions, add that there.  
 643

## 644 5 Open Challenges and Future Research Directions

### 645 5.1 Bayesian Optimization

646 5.1.1 *Open Challenges.*  
 647

648 5.1.2 *Future Directions.*  
 649

### 650 5.2 Low Rate Flow-table overflow attacks

651 5.2.1 *Open Challenges.*  
 652

653 Manuscript submitted to ACM  
 654

655

656

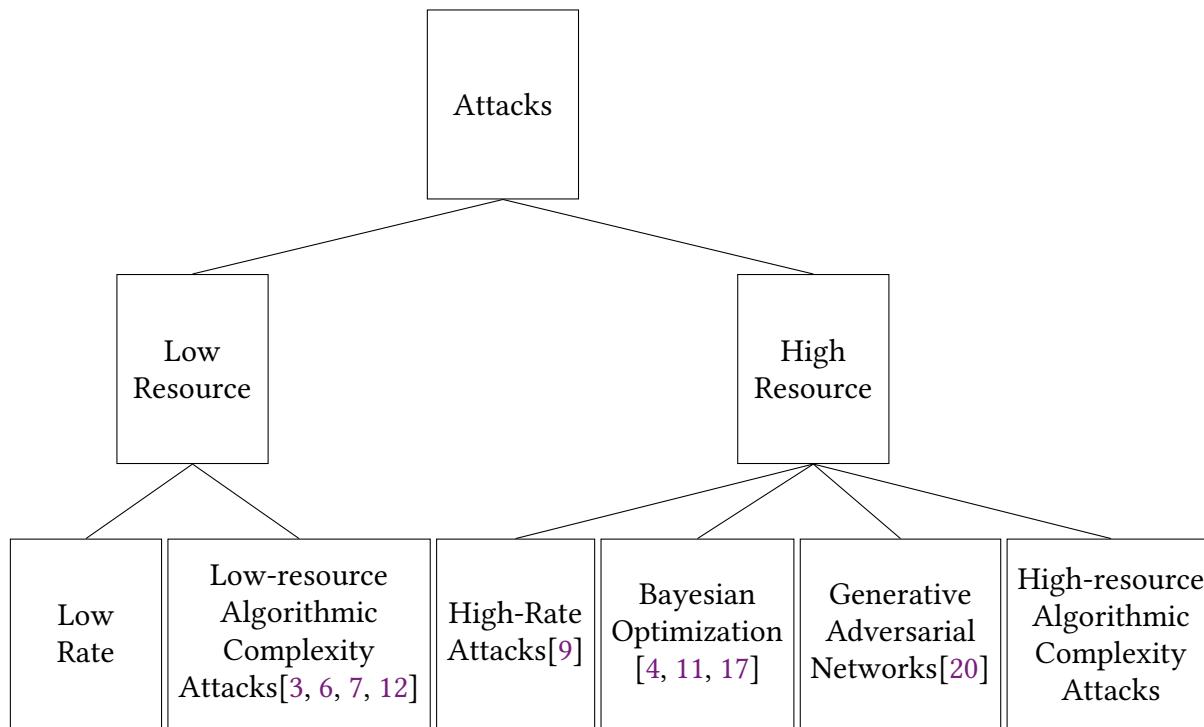


Fig. 3. Attack taxonomy

#### 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 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.

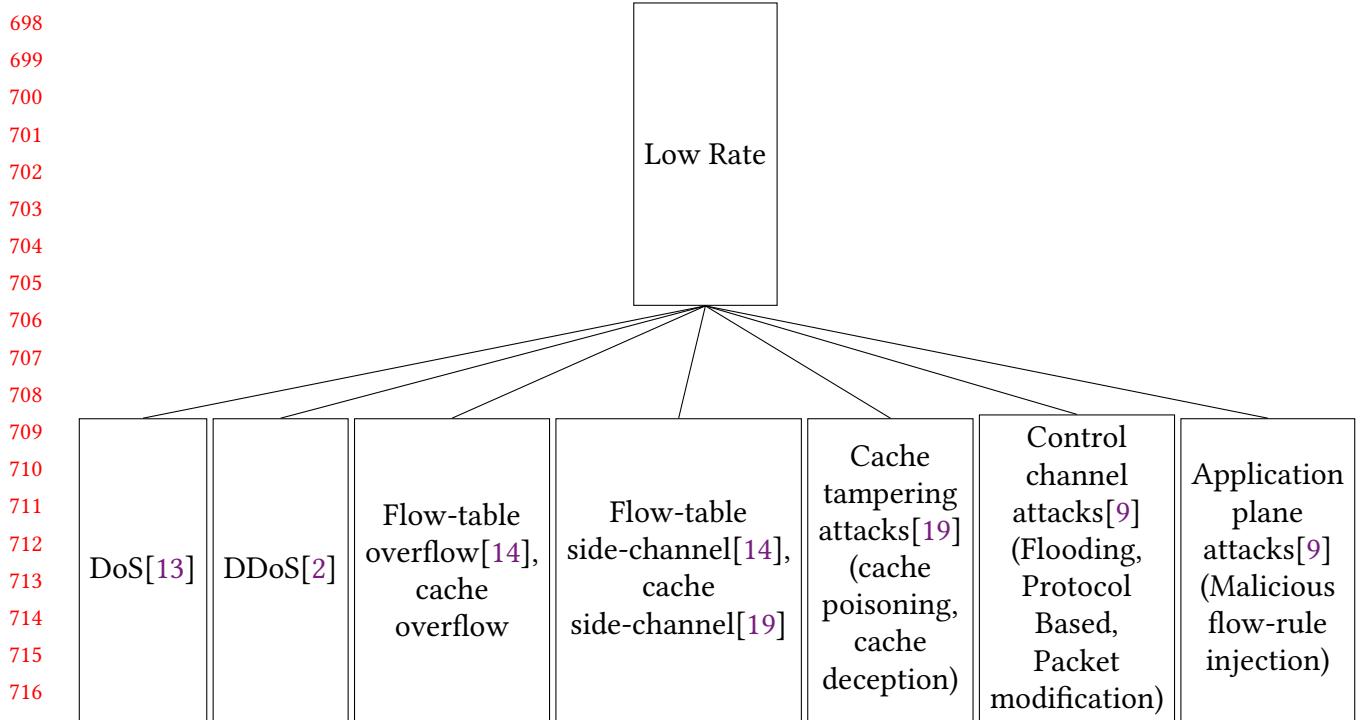


Fig. 4. Taxonomy of low-rate attacks

- [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 Ismaael 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 Kőrösí, 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 Fazzoli. 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](https://doi.org/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](https://doi.org/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