Zero-Click SnailLoad: From Minimal to No User Interaction

Stefan Gast[1]✉, Nora Puntigam[1], Simone Franza[1], Sudheendra Raghav Neela[1], Daniel Gruss[1], and Johanna Ullrich[2]

[1] Graz University of Technology, Austria
{stefan.gast,sudheendra.neela,daniel.gruss}@tugraz.at
{nora.puntigam,simone.franza}@student.tugraz.at
[2] University of Vienna, Austria
johanna.ullrich@univie.ac.at

Abstract. Network side channel often rely on privileged attacker positions, e.g., physical proximity, person-in-the-middle scenarios, or code on the victim machine. Recent fully remote attacks without a privileged position are either easily mitigated (e.g., ICMP pings) or require minimal user interaction (e.g., SnailLoad).
In this paper, we reduce user interaction in fully remote network side-channel attacks in two directions: First, we analyze 21 communication tools that automatically establish connections without user interaction with external references. We identify privacy-concerning automated behavior for 4 out of 11 messengers and 6 out of 10 email clients, leaking victim IP addresses without user interaction, undermining end-to-end encryption, and even enabling remote SnailLoad attacks without user interaction. Second, we show that even without any specific client software, merely processing TCP packets can already enable zero-click attacks. We introduce a novel latency measurement method based on TCP SYN packets, exploiting that TCP SYN packets to a closed port either lead to a timeout or, in our experiments about once per second, an ICMP-based response. Timing these responses yields a coarse SnailLoad trace, sufficient to mount a video fingerprinting attack with an F_1 score of 56 % compared to 89 % on a similar Internet connection and the same number of videos as prior work. Thus, our findings confirm in both directions that fully automated side-channel attacks without user interaction are feasible and posing a relevant privacy threat.

Keywords: Network side channel, Video fingerprinting, Timing

1 Introduction

Network side channels exploit observable features such as packet timings, sizes, the number of transmitted packets, and other metadata [28, 67, 39, 50, 19, 61, 17, 22, 60, 37, 2, 36], inferring sensitive information, including streamed videos [57, 20] or accessed websites [21, 73, 62, 3, 47, 10, 66, 58, 20], content from voice

communications [76], or sensitive personal details such as medical and financial information [14]. Traditionally, most attacks have relied on passive monitoring from within the same network or through attacker-controlled infrastructure along the communication path.

Only recently, some works have considered fully remote attack scenarios, where attackers measure latency exclusively from their own network packets [48, 46, 20]. For example, Murdoch and Danezis [48] demonstrated that network latency variations could reveal activities of a Tor relay node, while Gast et al. [20] showed how attackers could deduce victim website and video accesses solely from observed latency traces between attacker and victim. Fully remote attacks represent a significantly greater threat, as they require minimal attacker resources, can easily be scaled, and may be disguised within benign web content such as advertisements, enabling potentially widespread exploitation.

Despite being a greater threat, fully remote attacks typically still require some user interaction, e.g., clicking a link or visiting a webpage with attacker-controlled content. This limits realistic attack scenarios to e.g., malicious advertisements or some form of user interaction. Without the requirement of user interaction, attacks would be significantly easier to scale and mount in practice. Given many other contexts where remote resources are loaded, e.g., messenger and email programs, and some protocols requiring interaction with remote clients without initiating a connection, we ask the following research question: *Does system and software behavior enable fully remote network side-channel attacks like SnailLoad* ***without the need for user interaction?***

In this paper, we systematically investigate the possibility of fully remote network side-channel attacks ("zero-click") in two directions: the behavior of software and the network stack. **First**, we analyze realistic scenarios involving popular client-side software such as messenger platforms and email clients. Specifically, we examine whether such widely used applications automatically initiate external network connections upon receiving content, without explicit user interaction with the content (e.g., the message or email). In our experiments, we observe privacy- and confidentiality-compromising behaviors in 4 out of 11 messenger platforms and 6 out of 10 tested email clients. These applications automatically establish external connections, revealing victim IP addresses without any user interaction, undermining end-to-end encryption, and enabling zero-click remote SnailLoad-style attacks in practice.

Second, we consider an even more generic threat scenario. Even in absence of privacy- and confidentiality-compromising client software, the basic behavior of network stacks in common operating systems may enable zero-click attacks. To explore this, we develop a novel latency measurement method based solely on unsolicited TCP SYN packets sent from the attacker to a closed port on the victim's home gateway (*i.e.*, router). Unsolicited TCP SYN packets to closed ports trigger a *TCP RST* or an *ICMP Destination Unreachable* response, yielding a measurable latency for a SnailLoad-style attack. However, we observe that these requests often do not reach target systems or time out due to highly restrictive

rate limiting. Still, on 9 connections in our experiments, we are able to obtain a response to our unsolicited TCP SYN packets.

By distinguishing timings and ICMP responses of timed-out packets, we generate latency traces analogous to SnailLoad. However, our trace is significantly coarser, given the infrequent response rate of only one packet per second. While we expected this coarse granularity to be a hindrance for practical attacks, we discover that these sparse traces are in fact still sufficient to mount practical side-channel attacks. In a video fingerprinting attack with 10 videos, we achieve an F_1 score of 56 %, demonstrating the practicality of our zero-click method compared to SnailLoad's original F_1 score of 89 % on a similar Internet connection.

Our results clearly demonstrate that zero-click fully remote network side-channel attacks are not only theoretically possible but practical, highlighting significant privacy threats in everyday software and standard network behaviors. This expands the threat model of SnailLoad substantially beyond previous scenarios. Given that the vast majority of client systems use TCP/IP as well as messenger and email software today, this reveals a fundamental risk overlooked in prior work. Consequently, our work indicates that further mitigations need to be researched and deployed against fully remote network side-channel attacks, especially zero-click variants.

Contributions. We make the following key contributions:

1. We systematically analyze automated external resource handling across 11 messengers and 10 widely-used email clients. Our results reveal unsolicited connections in all 11 messengers, and 8 email clients, with 4 messengers and 6 email clients leaking the victim IP address.
2. We perform in-depth case studies for the affected messengers and 8 email clients. We identify 3 email clients vulnerable to single-click attacks and 4 messengers vulnerable to implicit attacks (triggered implicitly by other user actions), and 1 email client and 1 messenger to zero-click attacks.
3. We introduce a novel zero-click latency measurement approach based solely on unsolicited TCP SYN packets sent to closed ports on the victim gateway. Our experiments show that highly restrictive rate-limiting on the gateway typically only leads to a single response per second, which is orders of magnitude lower than the resolution from prior work.
4. We show that despite the coarse granularity of our latency measurements, our approach still suffices to mount practical video-fingerprinting attacks, reaching an F_1 score of 56 %, with zero user interaction.

Outline. Section 2 provides background. Section 3 systematically analyzes external reference handling in messenger and email clients. Section 4 introduces a novel TCP SYN-based measurement method for SnailLoad-style attacks. Section 5 discusses implications, mitigations, and limitations. Section 6 concludes.

2 Background

In this section, we provide background on network side channels including fully remote attacks, as well as automated behaviors in applications and protocols.

Network Side-Channel Attacks. Side-channel attacks exploit information leaking through measurable physical or behavioral characteristics of systems. Network side channels exploit packet timing [80, 19], packet sizes [27, 17, 58], transmission directions [74, 52, 5, 16, 34], packet data [8, 79, 40], or a combination thereof [55, 67, 11, 1, 77, 14, 51, 75, 76, 39, 4, 24, 50, 54, 56, 57, 42, 22, 64, 10, 65, 12, 36, 59, 23, 72]. Without needing direct access to the target, this allows adversaries to make inferences about the system state [8, 79, 40], a user [55, 1, 80, 75, 4], or their activities [28, 67, 14, 76].

Network side channels have been used to infer websites visited by a user [27, 26, 51, 74, 39, 24, 50, 19, 56, 61, 64, 10, 65, 58, 5, 16, 34, 59], videos streamed [57, 17, 52, 37, 2, 23, 72], applications used [39], and even sensitive data like medical records and voice conversations [14, 76]. Typically, these attacks operate in person-in-the-middle (PITM) scenarios, which require a significant degree of access or control over the victim's network infrastructure, such as being on the same (wireless) network or having access to a compromised router along the traffic path. Additionally, many attacks assume that the attacker can either capture packets directly or induce traffic on the victim's system through some form of code execution (e.g., JavaScript in a browser [1, 4, 57] or a native application [1, 75]). These assumptions restrict the practicality of deploying such attacks at scale, particularly against arbitrary victims on the open Internet.

Fully Remote Network Side-Channel Attacks. In contrast to traditional network side channels, fully remote variants do not rely on traffic interception or victim-side code execution [68, 7, 45, 35, 21, 44, 9, 20, 29]. With these, an off-path attacker exchanges network packets with a victim system, inferring sensitive data from observed responses. Eliminating the need for proximity or privilege, they are easier to mount and scale. While most of these attacks reconstruct properties of remote systems, e.g., their operating system or uptime, only two attacks focus, like the more powerful PITM-based attacks, on the identification of the accessed content by solely observing packet timing [21, 20]. Gong et al. [21] and Gast et al. [20] exploited contention at the bottleneck of standard internet connections – typically the last-mile link – where traffic queues cause measurable delays. Gong et al. [21] mounted a website-fingerprinting attack through round-trip times of ICMP Echo (*i.e.*, ping) messages to a victim's home router. Gast et al. [20] mounted website- and video-fingerprinting attacks through a TCP connection with the victim and measuring round-trip times of acknowledgments.

Unlike PITM attacks, fully remote network side-channel attacks require no special position or victim-side execution environment, aside from the requirement that the victim loads some asset from the attacker, e.g., an image or a background transfer. Many active hosts do not reply to ICMP requests [6, 29]. This mitigates the attack by Gong et al. [21], albeit at the expense of network diagnosis. Gast et al. [20] rely on TCP and cannot be mitigated in the same way. While powerful, the attack requires that the victim initiates a connection and loads some asset from the attacker, limiting deployment scenarios to content embedding or social engineering.

Security Risks of External Resources. Modern client applications, e.g., email and messenger platforms, and web browsers, routinely render external content, including images, style sheets, or fonts, fetched from third-party servers. In particular, email clients can fetch external resources referred to in HTML emails. Similarly, messenger applications often follow external links embedded into messages to display previews (e.g., for direct image links) or additional information (e.g., Open Graph [49] data for websites). This, however, introduces several security and privacy risks: First, accessing external resources may leak that a particular message has been read, e.g., tracking pixels or web bugs [18]. Second, additional metadata might be leaked such as the user's IP address, operating system and device behavior [79, 71]. Third, it enables attackers to inject or serve malicious content under the guise of legitimate-looking resources [25, 43, 69, 70].

For mitigation, applications employ proxy servers, sandboxing techniques, or prompt users before loading external content. However, these defenses are inconsistently applied and often disabled by users. For example, users might enable external content in their email client if mails render improperly otherwise. Moreover, even timing and volume of traffic via a proxy can still carry identifying information or enable fingerprinting [27, 78]. Attackers can deliver carefully crafted external references via email or messaging content, inducing connections to their own infrastructure without user consent. Kirchner et al. [38] found that 21 of 36 instant-messaging mobile apps and 20 of 41 web-based instant-messaging platforms analyze messages, visiting unique URLs and potentially leaking such side-channel information without user consent.

Such mechanisms can also activate background network interactions that serve as the basis for side-channel measurements. If external resources are fetched automatically upon message reception or preview, attackers may observe latency variations caused by concurrent victim activity. These behaviors, while subtle, introduce a broad attack surface for remote adversaries, especially when combined with fully remote side-channel techniques.

Automated Responses in Network Protocols. Network protocols such as TCP and ICMP include built-in mechanisms for responding to unsolicited packets. These behaviors are crucial for the proper functioning of the internet, yet they also represent a potential side-channel vector. In particular, responses to malformed, unexpected, or incomplete packets can expose timing information that reveals network congestion, bandwidth bottlenecks, or the presence of specific services [29]. For example, TCP initiates connections using a three-way handshake beginning with a SYN packet [33]. If a SYN is sent to an open port, a SYN-ACK is returned; if the port is closed, the system may respond with a TCP RST or an ICMP *Destination Unreachable* message. ICMP itself is designed as a diagnostic protocol, with messages such as *Echo Reply*, *Time Exceeded*, and *Port Unreachable* intended to inform senders of network conditions [30, 31].

Modern operating systems implement rate-limiting and filtering to restrict volume and frequency of responses. For example, ICMP Echo (*i.e.*, ping), which sends a ping request to check if a host is reachable, is often disabled for security [41, 63, 13]. Additionally, ICMP rate limiting may suppress replies to other

repeated probes, and firewalls may drop unsolicited packets altogether. However, in practice, many devices still emit rate-limited responses, e.g., on home internet gateways (*i.e.*, home routers), given their verbose default configurations [29].

3 Exploiting Embedding of External Elements

Gast et al. [20] achieved client connections to an attacker-controlled server via an external reference embedded in a benign website the victim user visits. While requiring no elevated attacker privileges or code execution, this model still depends on user interaction, e.g., clicking a link or loading a webpage.

In this section, we systematically analyze how far user interaction can be eliminated with default-configured native and web-based messenger and email applications. We evaluate whether these applications automatically initiate network connections upon receiving, processing, or displaying references to external references. Such behavior would allow a fully automated version of SnailLoad, bypassing the user entirely, which we call *Zero-Click SnailLoad*.

Threat Model. We assume an attacker able to legitimately send messages or emails to the victim. The attacker has no further access to the victim's system, no ability to execute code, and no control over the underlying network infrastructure. The attacker-controlled server hosts a file or image referenced via a URL. The victim receives a message or email referencing the attacker's server. The attacker aims to mount a SnailLoad attack, without any interaction by the victim. We consider two attack vectors: The first attack vector is link previews, message rendering, or other background processing in **messenger clients**. The second attack vector is external references in HTML emails in **email clients**.

For completeness, we also investigate the possibility of attacking the sender of a message: In a scenario where the sender forwards a link to a benign-looking image (e.g., a meme), that actually attacks the sender using SnailLoad. In this scenario, the victim does not click or open the link but only forwards it. This also undermines end-to-end encryption as parts of the message are used to trigger a request to a remote server of either the attacker or the platform provider.

3.1 Evaluation Methodology

We deploy a SnailLoad-style server that records information of the potential unsolicited network connections. The server logs the following information for each incoming connection: (1) the source IP address, to determine whether it is the victim's client connecting, a proxy or a web cache by the messenger platform and (2) whether the external resource is loaded or the connection aborts. To the outside, the external resource appears as a static file. We embed the external reference into each message or email, e.g., in the form of a simple image tag: `<img src="http://atta.ck/plot.jpg">`. The image may or may not be visible on the client, e.g., a legitimate-looking image or a small tracking pixel.

We evaluate whether the client opens a connection (1) immediately upon message receipt, e.g., email delivery on the client; (2) when there is interaction

Table 1. Observed behavior of messenger platforms with embedded external references

Messenger	Victim Role	Trigger	Link Image Previews	Link Website Previews	Client / Server Connection	Caching	Latencies Observable	Implicit Attack
Discord	Sender	Send	✓	✓[1]	Server	✓[2]	✓[3]	✗
	Receiver	–	✓[4]	✓[4]	–	✓[2]	✗	✗
Facebook Mes.	Sender	Send	✓	✓[1]	Server	✓[5]	✓[3]	✗
	Receiver	–	✓[4]	✓[4]	–	✓[5]	✗	✗
Google Chat	Sender	Type, Send	✓	✓[1]	Server	✓[5]	✓[3]	✗
	Receiver	–	✓[4]	✓[4]	–	✓[5]	✗	✗
iMessage[6]	Sender	Type, Send	✓	✓[1]	Server	✗	✓[3]	✗
	Receiver	–	✓[4]	✓[4]	–	✗	✗	✗
iMessage[7]	Sender	Type, Send	✓	✓[1]	Client	✗	✓	✓
	Receiver	–	✓[4]	✓[4]	–	✗	✗	✗
Instagram	Sender	Send	✓	✓[1]	Server	✓[5]	✓[3]	✗
	Receiver	–	✓[4]	✓[4]	–	✗	✗	✗
Microsoft Teams	Sender	Send	✓	✓[1]	Server	✓[5]	✓[3]	✗
	Receiver	Open Chat	✓	✓[1]	Server	✓[5]	✓[3]	✗
Signal	Sender	Type	✗	✓[1]	Client	✗	✓	✓
	Receiver	–	✗	✓[4]	–	✗	✗	✗
Snapchat	Sender	Send	✓	✓[1]	Server	✓[5]	✓[3]	✗
	Receiver	–	✓[4]	✓[4]	–	✗	✗	✗
Telegram	Sender	Type, Sending	✓	✓[1]	Server	✓[2]	✓[3]	✗
	Receiver	–	✓[4]	✓[4]	–	✓[2]	✗	✗
Viber	Sender	Send	✓	✓[1]	Client	✓[8]	✓	✓
	Receiver	Open Chat	✓[9]	✓[1,9]	Client[9]	✓[8,9]	✓[9]	✓[9]
Whatsapp	Sender	Type	✓	✓[1]	Client	✓[8]	✓	✓
	Receiver	–	✓[4]	✓[4]	–	✓	✗	✗

[1] also follows `og:image` Open Graph meta tag, [2] server-side, only main link cached, `og:image` is followed every time, [3] only server-side latencies, [4] same preview as sender, [5] server-side, [6] iCloud Private Relay enabled, [7] iCloud Private Relay disabled, [8] client-side, until app is closed, [9] only if sender is in contact list

with other messages, e.g., scrolling or hovering over the malicious message; (3) when clicking the message and it is being rendered; and for completeness also (4) when entering or sending a link. Each test is repeated multiple times per platform to ensure reproducibility and to assess caching behavior.

3.2 Messenger Platforms

Messenger platforms are an attractive target for remote side-channel attacks due to their ubiquity, always-on behavior, and support for rich message content, including embedded links. To assess their vulnerability to zero-click SnailLoad-style attacks, we systematically analyzed 11 popular messengers: Discord, Facebook Messenger, Google Chat, iMessage, Instagram Direct Messenger, Microsoft Teams, Signal, Snapchat, Telegram, Viber, and WhatsApp. Each platform is

tested using its default configuration across desktop, web, and mobile variants (if they exist). For iMessage, we measure two configurations, one with iCloud Private Relay enabled and one without. We largely find the same behavior across platforms, with similar backend handling of messages (e.g., through a CDN).

For each platform, we examine two types of referencing an attacker-controlled server: a direct link to an image (e.g., `http://atta.ck/plot.jpg`) and a link to a website (e.g., `http://atta.ck`). The website also has an Open Graph `og:image` meta tag, used by all messengers to retrieve website preview images, potentially also enabling attacks. This allows us to observe the connection behavior for both types of links, and potential differences, in various usage scenarios. More specifically, we test whether a connection is established when composing a message, sending it, interacting with the app but not the conversation, opening the conversation, or interacting with the embedded link. We record whether a client-side or server-side connection occurred, whether the user's IP address was exposed, whether repeated interactions triggered additional requests (indicating lack of caching), and whether a link preview was rendered. Finally, we determined whether these conditions suffice to enable a zero-click or implicit variants of SnailLoad, which we only found in 4 messengers.

The results, summarized in Table 1, show that all messengers perform some form of automatic interaction with external links, establishing unsolicited connections, typically through server-side proxies. Concretely, 8 out of 11 platforms initiate server-side connections when messages are sent or received. Note that this undermines end-to-end encryption as parts of the message, *i.e.*, the URL, are exposed to the platform provider without user consent.

We observe that 4 messengers, Whatsapp, Signal, iMessage and Viber, expose user IP addresses through direct client-side connections. For these messengers, a connection is triggered immediately upon typing the link or when sending the final message, and is made directly from the sender's device. This exposes not only when a link is sent but also enables an attacker to target a victim forwarding a benign-looking image to others. While caching may limit the effect, unique links can be sufficient to identify users. For iMessage, the connection behavior is more nuanced: The user's IP address is exposed when sending a link to an external resource but only upon receiving a message if iCloud Private Relay is disabled. Otherwise, the IP address is not exposed but also no manual interaction is required for the proxy to access the resource. Thus, iMessage enables an implicit SnailLoad attack, *i.e.*, no user interaction with the external resource, on the sender and, in some configurations, also on the receiver. For Viber, we found the behavior even more security- and privacy-concerning: In addition to opening a client-side connection when sending the message, we also observed it to open a client-side connection when the receiving user opens the chat window and has the sender in the contact list, enabling targeted SnailLoad attacks.

We conclude that exploiting unsolicited link interactions for SnailLoad-type attacks is a feasible attack vector for some messengers. Most messengers use proxy servers for interaction with the remote server, which still leaks side-channel information and in particular undermines the confidentiality of the communica-

tion channel through out-of-band requests to the platform provider, e.g., undermining end-to-end encryption. However, messengers vulnerable to zero-click and implicit attacks using no proxy server enable full SnailLoad attacks, *i.e.*, they raise stronger privacy concerns.

3.3 Email Clients

Email clients are also an attractive target for remote side-channel attacks. They are widely used, run most of the time, and support embedding of external resources, e.g., pictures. To assess their vulnerability to zero-click SnailLoad-style attacks, we systematically analyze 10 email clients: Apple Mail (macOS, iPhone, Watch), BlueMail, eM Client, Gmail, GMX, Mailbird, Microsoft Outlook, Proton Mail, Spark Mail, and Thunderbird. Each platform is tested using its default configuration across desktop, web, and mobile variants (if they exist). For Apple Mail, we explicitly test different configurations of *Protect Mail Activity* and *iCloud Private Relay* to assess privacy-relevant differences. We specify when applications are found to expose different behavior on different platforms.

We embed a static reference to an image on an attacker-controlled server into an HTML email and observe connection behavior in various usage scenarios. We focus only on the receiver side as the client email was crafted by the attacker. We record whether external references are loaded automatically, what kind of connection is established (client-side or via a proxy), whether the user's IP address is exposed, and which user actions (if any) triggers the connection. We record whether a client-side connection occurs and whether the image is automatically rendered. Finally, we determine whether these conditions suffice for zero-click SnailLoad, which we found in the Spark Mail email client.

Table 2 summarizes the connection behavior of all tested email clients. Compared to messengers, where most platforms use a proxy to access the image, we observe a much higher rate of direct client-side IP exposure in email clients. Across all tested clients, **6 out of 10** establish client-side connections. In *BlueMail*, the connection is made automatically upon opening the email, in *Spark Mail* even upon just hovering the email. In particular the behavior of *Spark Mail* is critical as it enables an end-to-end zero-click SnailLoad attack. Apple Mail shows different behavior across devices. On macOS and iPhone with *Protect Mail Activity* and *iCloud Private Relay* enabled, external content is retrieved through proxies, e.g., Cloudflare. In contrast, Apple Mail on the Apple Watch establishes a direct connection if the email has previously been opened on another device, making it vulnerable to a time-delayed attack scenario. This inconsistency in behavior can introduce privacy compromises that may go unnoticed by users who assume Apple Mail behaves uniformly across devices. BlueMail exposes similar behavior in all scenarios when opening an email. Web-based clients expose less side-channel information in our attack. Proton Mail and GMX block all unsolicited external connections and require explicit interaction for the remote resource to load. Gmail, while automatically loading external references, consistently uses proxy infrastructure. While this masks the user's IP address, it still leaks when the user opens an email, which can be a privacy concern.

Table 2. Observed behavior of email clients with embedded external references

Client	Trigger	Client Conn.	Server Conn.	Autom. Display	Caching	Single-Click Att.	Zero-Click Att.
Apple Mail (macOS)	Open	✗	✓	✓	✓	✗	✗
Apple Mail (iPhone)	Open	✗	✓	✓	✓	✗	✗
Apple Mail (Watch)	Open	✓	✗	✓	✓	✓	✗
BlueMail	Open	✓	✗	✓	✗	✓	✗
eM Client	Manual Click	✓	✗	✗	✓	✗	✗
Gmail (Web/iPhone)	Open	✗	✓	✓	✓	✗	✗
GMX (Web)	None	✗	✗	-	-	-	-
Mailbird	Manual Click	✓	✗	✗	✓	✗	✗
Microsoft Outlook	Manual Click	✗	✓	✗	✓	✗	✗
Proton Mail (Web)	None	✗	✗	-	-	-	-
Spark Mail	Hover Inbox	✓	✗	✗	✗	✓	✓
Thunderbird	Manual Click	✓	✗	✗	✓	✗	✗

Similarly, Microsoft Outlook relies on user-initiated clicks and proxy loading, exposing similar information as Gmail albeit upon a user click.

In summary, web-based clients behave more similar to messengers, where server-side connections are the default, whereas native email clients and apps frequently expose user metadata to remote servers via client-side connections. BlueMail and Apple Mail on the Apple Watch require the user to open the email to run a SnailLoad attack. Spark Mail enables reliable zero-click SnailLoad attacks without even opening the email.

4 Eliminating User Interaction with TCP SYNs

In the previous section, we minimized user interaction significantly, yet some level of interaction remained unavoidable. This leads to the continued exploration of the paper's central question: Can user interaction be entirely eliminated?

To address this, the threat model pivots from a direct TCP connection with the victim to an attacker interacting with the victim's home gateway. As ICMP Echo Replies are frequently disabled and TCP ports closed, we explore whether the router's responses to unsolicited TCP SYNs are another source of network latency leakage. Even if a connection cannot be established, the router may respond with a TCP RST or an ICMP Destination Unreachable that can be leveraged to gather network activity patterns. First, we examine 4 different internet connection and home gateway combinations to analyze under which conditions their response behavior can be exploited. Second, we mount a video fingerprinting attack, achieving an F_1 score of 56 % for 10 YouTube videos, showing that

Table 3. Requirements for the TCP SYN attack. The victim needs to have a public IP address (without carrier-grade NAT), with the home gateway responding to incoming TCP SYN packets on the closed port. Rate limits on the home gateway only reduce the attacker's effective sample rate, but cannot prevent the attack.

Carrier-Grade NAT	●	●	●	●	○	○	○	○
Home Gateway Responding	●	●	○	○	●	●	○	○
Rate Limit	●	○	●	○	●	○	●	○
Attack feasible	✗	✗	✗	✗	✓	✓	✗	✗

such attacks are feasible even without user interaction. Third and finally, to assess the practicality on other connections, we perform a user study with the internet connections of 102 participants, revealing that 9 of the examined connections are potentially exploitable.

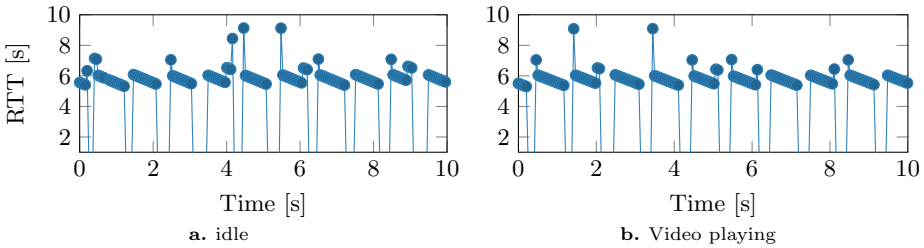
4.1 Examining TCP SYN Response Behavior

In this section, we investigate whether closed port TCP SYN response timings can be generally used to infer a victim's network activity. Therefore, we initially observe the behavior of 4 Internet connection and home gateway combinations. From the observed behavior, we derive the prerequisites for an attack.

Measurement Setup. We send TCP SYN packets with a specific destination port, to the gateway's public IP address and record the round-trip times between the sent TCP SYN and the received response (or an encountered timeout). For each tested Internet connection and gateway, two sets of measurements are performed. First, latency traces are recorded while the network remains idle to establish a baseline. Then, we record traces while a video stream is playing on YouTube to observe potential differences in response behavior. We repeat each test multiple times with varying TCP SYN intervals ranging from 50ms to 100ms, on multiple ports to ensure the observed effects are not port-specific.

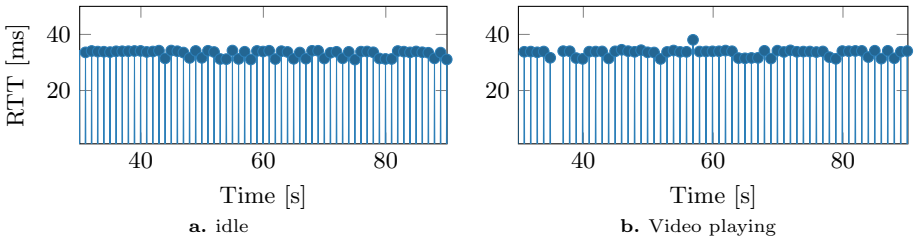
Results and Key Observations. We examine 4 connection and gateway combinations at two different ISPs. We observe different behaviors across the four tested connections, revealing varying levels of susceptibility to the attack:

For both cable connections from *ISP A*, we did not receive any responses from both of the two tested routers. This behavior suggests that these routers do not respond in a way that could reveal network activity and that the attack is not feasible for these routers. For the cable connection from *ISP B*, we receive responses to our TCP SYN packets, see Figure 1. Interestingly, the router only replies after receiving multiple TCP SYN packets, answering the packets accumulated until that point in rapid succession, causing rather large baseline response delays. Instead of consistently receiving responses, the results show a mix of refusals and timeouts occurring in evenly spaced intervals, suggesting a token bucket-based rate limiter. Comparing Figures 1a and 1b, the pattern is consistent for the idle network and during video streaming. This pinpoints the



a. idle **b.** Video playing

Fig. 1. TCP SYN response times of a 120 Mbit/s cable connection, idle and with a video playing, with a 10 s timeout for responses. We only receive responses after sending multiple TCP SYN packets, which then are replied to in rapid succession, causing a sawtooth pattern. Additionally, packets are silently dropped in periodic intervals, indicating a rate-limit. The traces for an idle and a busy connection look very similar, as the TCP SYN packets are handled by the carrier-grade NAT and therefore do not travel over the last-mile bottleneck.



a. idle **b.** Video playing

Fig. 2. TCP SYN response times of a 50 Mbit/s ADSL connection, idle and with a video playing, with a 50 ms timeout for responses. When idle, we receive a response once per second within approximately 35 ms. When a video is playing, we observe increased response times and additional timeouts when the client is buffering a video segment.

presence of carrier-grade NAT [32], *i.e.*, multiple home gateways share a public IPv4 address. As a result, TCP SYN packets sent to the victim's external IP address do not reach their home gateway but are instead handled at the ISP level. Consequently, the attack is not applicable in this scenario.

In contrast, the DSL connection from *ISP B* exhibits two clearly distinct and reproducible patterns, depending on whether the network is idle or actively streaming a video. As shown in Figure 2, the latency traces for an idle network follow a stable pattern, whereas measurements taken during video streaming display a unique and consistently observable fluctuation. This effect is present across all measurement variations, confirming that the attack successfully detects network activity.

Evaluation. Our results demonstrate that closed port TCP SYN responses can indeed be exploited to measure network latencies and to capture distinct patterns of network activity. This offers a potential solution to eliminate the user interaction element in SnailLoad. However, it is evident that the attack is not universally applicable to all routers or ISPs. In particular, two conditions

must be met for the attack to be successful: First, the victim's gateway must have a public IP address, without relying on carrier-grade Network Address Translation. Otherwise, the requests are handled by the victim's ISP and do not reach the internet connection's last mile, a prerequisite of the attack. Second, the victim's home gateway has to respond to incoming TCP SYNs designated to a closed port. These responses often are rate-limited, yet, in Section 4.2 we show that this does not prevent attacks. Table 3 summarizes these prerequisites.

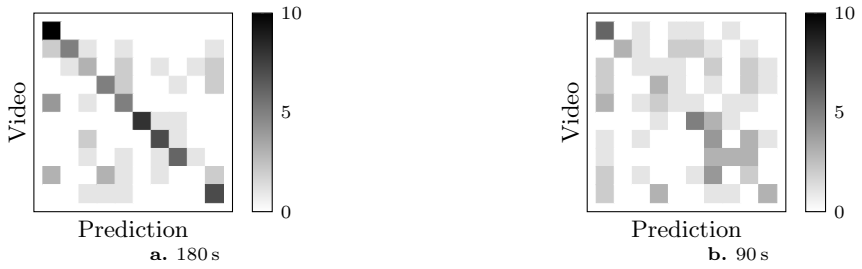
4.2 Video Fingerprinting with TCP SYNs

In this section, we mount an end-to-end video fingerprinting attack, exploiting gateway responses to TCP SYN packets sent to a closed port. While the victim user streams a video, the attacker repeatedly sends TCP SYN packets and records the latencies of the corresponding ICMP Destination Unreachable messages sent by the gateway. We demonstrate our attack on a set of 10 videos trending on YouTube. For each video, we record 50 latency traces of 180 s length in Full HD. We then train a CNN on 40 of the traces collected for each video and evaluate the attack on the remaining 10 traces. Our attack achieves an F_1 score of 56 %, significantly higher than the random guessing accuracy of $F_1 = 10\,\%$.

Threat Model. Like in Section 3, the attacker is unable to intercept any traffic and cannot run any code on the victim's system. We assume the victim's gateway does not respond to ICMP echo messages. However, we assume the router responds to incoming TCP SYN packets destined to closed ports with either TCP RST or ICMP Destination Unreachable messages. Additionally, we consider the victim's gateway to have a public IP address without carrier-grade NAT (cf. Section 4.1), and that the attacker knows the victim's IP address.

Trace Recording. Our setup consists of the victim clien, playing the YouTube videos, and an attacker-controlled virtual server in the cloud, recording the latency traces. The client is connected to the home gateway, which is connected to the Internet via a 50 Mbit/s ADSL connection. The home gateway is in its default configuration, rate limiting ICMP to 1 packet per second. A script on the victim client controls the automated recording of the individual traces. We record 50 traces per video, *i.e.*, 500 traces in total, recorded in random order to prevent order-related effects. For each trace, the script signals the start of a new trace to the server, waits for 1 s to 3 s and starts playing the video in Firefox 136.0.3. While the video is playing, the server repeatedly sends TCP SYN packets to a closed port on the victim's gateway, in 50 ms intervals. For each TCP SYN packet, the server records the latency of the corresponding response. For unresponded TCP SYNs, the server records a latency of 50 ms. After 180 s, the script signals the end of the trace to the server and stops the video.

Training the CNN. Like prior work [15, 53, 20], we apply a Short-Time Fourier Transform (STFT) to each trace and train a Convolutional Neural Network (CNN) on the results. Due to the ICMP rate limit, we choose a rather large window size of 256 samples for the STFT, *i.e.*, 12.8 s with a sampling interval of 50 ms. Our CNN consists of 3 convolutional layers, each of which followed by a max pooling layer and a dropout layer. These 3 blocks are followed by a flatten



a. 180 s **b.** 90 s

Fig. 3. Confusion matrices of the video-fingerprinting attack based on TCP SYN packets. On a 50 Mbit/s ADSL connection, we achieve an F_1-score of 56 %, with 50 traces per video and a recording time of 180 s per trace. For comparison, if we reduce the recording time to 90 s, the F_1-score drops to 31 %.

layer and another dropout layer. This dropout layer is followed by 2 blocks, each consisting of a dense layer and a dropout layer. These blocks are followed by a final dense layer, yielding the estimated likelihoods that the input corresponds to the specific label for each possible label. Out of the 50 traces for each video, we use 36 traces as a training set to fit the model, 4 traces as a validation set to evaluate the model during training and 10 traces as a test set for the evaluation of the trained model. We empirically choose the hyperparameters for the CNN to achieve good generalization against the validation set and apply early-stopping to reduce overfitting.

Results. We evaluate the trained model on the test set, consisting of the remaining 10 traces per video. Despite the rate-limit imposed by the victim's gateway, we achieve an F_1-score of 56 %, significantly higher than random guessing (which would be $F_1 = 10\,\%$). On a similar ADSL connection, Gast et al. [20] reported a higher accuracy of $F_1 = 89\,\%$ for their attack based on TCP ACKs, with a recording time of only 90 s per trace. However, their video-fingerprinting attack worked with a sample rate of 20 Hz, while we are effectively restricted to only 1 Hz by the ICMP rate-limit. For comparison, we also evaluate our attack with the recorded traces truncated to 90 s. With this, the accuracy drops to only $F_1 = 31\,\%$. Figure 3 shows the confusion matrices for both 180 s and 90 s traces. Our results show that rate-limiting is not sufficient to mitigate leakage from response timings. Even with a relatively strict rate-limit of only 1 response per second, video-fingerprinting attacks are still feasible.

4.3 User Study

For an estimate of the general applicability of this attack, we conducted a user-study with 102 computer science students. With their consent, we collected their IP addresses and examined the response behavior of the IP addresses with regard to Section 4.1. We also asked the participants about the type of Internet connection they have and whether they have a true public IP address, without a carrier-grade NAT. Figure 4 summarizes the results of our study, finding 9 out

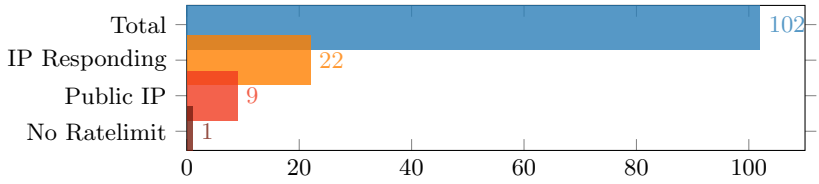


Fig. 4. User study results on internet connections with 102 participants. For 22 connections responded to TCP SYN requests to closed ports. Of these 22 connections, 9 had a public IP without carrier-grade NAT, making them potentially vulnerable to our attack. 1 of the vulnerable connections did not even rate limit the responses.

of 102 (*i.e.*, 8.8 %) of the examined connections to be potentially vulnerable. 7 of them responded with ICMP Destination Unreachable messages, all of them rate-limited to 1 response per second. 1 home gateway responded with TCP RST messages, delayed in a similar way as the responses from the cable connection in Section 4.1. 1 home gateway responded with TCP RST messages and did not even rate-limit them.

5 Discussion

The feasibility of fully automated, zero-click network side-channel attacks depends strongly on the behavior of applications and the network. Our work shows that there is a significant variance across messengers and email clients in their default behavior and with how little interaction they expose users to this threat, e.g., single- or zero-click. Still, our results are sufficient to show that it is not necessary for the victim to initiate a connection through user interaction, such as opening an email or opening a website. This does not only make the attack by Gast et al. [20] more scalable, with little to no user interaction, it also exposes victim timing information and IP addresses. Surprisingly, we also found that both client-side proxied server-side accesses pose a separate privacy risk as these accesses undermine end-to-end encryption by exposing message parts, *i.e.*, text looking like a URL, to out-of-band provider APIs, e.g., to generate previews. There is an abundance of applications beyond the ones we investigated that could similarly serve as targets of our attacks.

Similarly, our TCP SYN-based measurements depend on the handling of TCP SYN requests. As described in Section 4, many networks simply ignore these requests and, thus, do not expose themselves to zero-click SnailLoad-style attacks. Still, a significant number of users is affected, rendering our attack a real-world threat. With increasing adoption of IPv6 for private internet connections, the need for NAT decreases, exposing more devices behind the last-mile bottleneck as potential targets. Furthermore, other protocols may expose similar behavior that leaks round-trip time information. In essence, with our work, the SnailLoad attack surface has shifted from attacks depending on *some* user activity to unknowingly initiate the attack to attacks where the user does not initiate the attack, cannot see that an attack is going on, nor interfere with the

attack. Consequently, SnailLoad-style attacks must be taken more seriously and mitigations are needed for commodity systems.

6 Conclusion

In this paper, we systematically investigated realistic scenarios for fully remote network side-channel attacks requiring no user interaction. We analyzed software-induced and network-level vectors, showing that attackers can eliminate the need for explicit user action while still mounting practical attacks.

First, we examined the handling of external references in widely-used software. Our analysis of 10 email clients and 11 messenger platforms revealed privacy- and confidentiality-compromising behavior in 6 email clients and 4 messengers. 4 messengers and 1 email client automatically initiate external connections from the client machine, undermining end-to-end encryption, enabling attackers to exfiltrate victim IP addresses and to perform SnailLoad-style attacks.

Second, we introduced a novel TCP SYN-based latency measurement approach that bypasses software-specific behavior entirely. By sending TCP SYN packets to closed ports, we observe a low-rate but consistent signal of ICMP responses (about 1 packet per second). Despite this coarseness, we demonstrate the feasibility of video fingerprinting attacks. This shows that even fundamental behaviors of the network stack enable zero-click remote attacks, independent of specific application-level functionality.

Taken together, our results show that zero-click fully remote side-channel attacks are not only theoretically possible but practically feasible. They expose a previously underestimated attack surface, arising from both common software practices and inherent network behavior. This highlights an urgent need for reconsidering assumptions in the design of secure systems and network stacks.

Acknowledgments

The final version of this paper is published at ESORICS 2025. We thank our anonymous reviewers for their valuable feedback. This research is supported in part by the European Research Council (ERC project FSSec 101076409), the Austrian Science Fund (FWF SFB project SPyCoDe 10.55776/F85 and FWF project NeRAM 10.55776/I6054), and SBA-K1 NGC, a COMET Center within the COMET – Competence Centers for Excellent Technologies Programme, funded by BMIMI, BMWET, and the federal state of Vienna. Additional funding was provided by a generous gift from Intel. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding parties.

References

1. Abbot, T., Lai, K., Lieberman, M., Price, E.: Browser-Based Attacks on Tor. In: PET (2007)

2. Afandi, W., Bukhari, S.M.A.H., Khan, M.U.S., Maqsood, T., Khan, S.U.: Fingerprinting Technique for YouTube Videos Identification in Network Traffic. IEEE Access **10**, 76731–76741 (2022)
3. Apthorpe, N., Reisman, D., Sundaresan, S., Narayanan, A., Feamster, N.: Spying on the smart home: Privacy attacks and defenses on encrypted iot traffic. arXiv:1708.05044 (2017)
4. Arp, D., Yamaguchi, F., Rieck, K.: Torben: A Practical Side-Channel Attack for Deanonymizing Tor Communication. In: ASIA CCS (2015)
5. Bahramali, A., Bozorgi, A., Houmansadr, A.: Realistic Website Fingerprinting By Augmenting Network Traces. In: CCS (2023)
6. Bano, S., Richter, P., Javed, M., Sundaresan, S., Durumeric, Z., Murdoch, S.J., Mortier, R., Paxson, V.: Scanning the Internet for Liveness. In: Computer Communications Review (2018)
7. Bender, A., Sherwood, R., Spring, N.: Fixing Ally's Growing Pains with Velocity Modeling. In: SIGCOMM (2008)
8. Beverly, R.: A Robust Classifier for Passive TCP/IP Fingerprinting. In: PAM (2004)
9. Beverly, R., Luckie, M., Mosley, L., Claffy, K.: Measuring and Characterizing IPv6 Router Availability. In: PAM (2015)
10. Bhat, S., Lu, D., Kwon, A., Devadas, S.: Var-CNN: A Data-Efficient Website Fingerprinting Attack Based on Deep Learning. PoPETS (2019)
11. Bissias, G.D., Liberatore, M., Jensen, D., Levine, B.N.: Privacy Vulnerabilities in Encrypted HTTP Streams. In: PET (2006)
12. Bushart, J., Rossow, C.: Padding Ain't Enough: Assessing the Privacy Guarantees of Encrypted DNS. In: USENIX FOCI (2020)
13. Chaba, Y., Singh, Y., Aneja, P.: Performance Analysis of Disable IP Broadcast Technique for Prevention of Flooding-Based DDoS Attack in MANET. Journal of Networks **4**(3), 178–183 (2009)
14. Chen, S., Wang, R., Wang, X., Zhang, K.: Side-Channel Leaks in Web Applications: A Reality Today, a Challenge Tomorrow. In: S&P (2010)
15. Chen, Z., Xu, Y.Q., Wang, H., Guo, D.: Deep STFT-CNN for spectrum sensing in cognitive radio. IEEE Communications Letters (2020)
16. Deng, X., Yin, Q., Liu, Z., Zhao, X., Li, Q., Xu, M., Xu, K., Wu, J.: Robust Multi-tab Website Fingerprinting Attacks in the Wild. In: S&P (2023)
17. Dubin, R., Dvir, A., Pele, O., Hadar, O.: I Know What You Saw Last Minute—Encrypted HTTP Adaptive Video Streaming Title Classification. IEEE Transactions on Information Forensics and Security **12** (2017)
18. Englehardt, S., Han, J., Narayanan, A.: I never signed up for this! Privacy implications of email tracking. In: PETS (2018)
19. Feghhi, S., Leith, D.J.: A Web Traffic Analysis Attack Using Only Timing Information. IEEE Transactions on Information Forensics and Security (2016)
20. Gast, S., Czerny, R., Juffinger, J., Rauscher, F., Franza, S., Gruss, D.: SnailLoad: Exploiting Remote Network Latency Measurements without JavaScript. In: USENIX Security (2024)
21. Gong, X., Borisov, N., Kiyavash, N., Schear, N.: Website Detection Using Remote Traffic Analysis. In: PETS (2012)
22. Gu, J., Wang, J., Yu, Z., Shen, K.: Walls Have Ears: Traffic-based Side-Channel Attack in Video Streaming. In: INFOCOM (2018)
23. Hasselquist, D., Witwer, E., Carlson, A., Johansson, N., Carlsson, N.: Raising the Bar: Improved Fingerprinting Attacks and Defenses for Video Streaming Traffic. PoPETS (2024)

24. Hayes, J., Danezis, G.: k-fingerprinting: A Robust Scalable Website Fingerprinting Technique. In: USENIX Security (2016)
25. Heiderich, M., Niemietz, M., Schuster, F., Holz, T., Schwenk, J.: Scriptless attacks: stealing the pie without touching the sill. In: CCS (2012)
26. Herrmann, D., Wendolsky, R., Federrath, H.: Website Fingerprinting: Attacking Popular Privacy Enhancing Technologies with the Multinomial Naïve-Bayes Classifier. In: CCSW (2009)
27. Hintz, A.: Fingerprinting Websites Using Traffic Analysis. In: PET (2003)
28. Hogye, M.A., Hughes, C.T., Sarfaty, J.M., Wolf, J.D.: Analysis of the Feasibility of Keystroke Timing Attacks over SSH Connections. Tech. rep., School of Engineering and Applied Science University of Virginia (2001)
29. Holzbauer, F., Maier, M., Ullrich, J.: Destination Reachable: What ICMPv6 Error Messages Reveal About Their Sources. In: IMC (2024)
30. Internet Engineering Task Force: RFC 792: Internet Control Message Protocol (1981), https://datatracker.ietf.org/doc/html/rfc792
31. Internet Engineering Task Force: RFC 4443: Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification (2006), https://datatracker.ietf.org/doc/html/rfc4443
32. Internet Engineering Task Force: Common Requirements for Carrier-Grade NATs (CGNs) (2013), https://datatracker.ietf.org/doc/rfc6888/
33. Internet Engineering Task Force: RFC 9293: Transmission Control Protocol (TCP) (2022), https://datatracker.ietf.org/doc/html/rfc9293
34. Jin, Z., Lu, T., Luo, S., Shang, J.: Transformer-based Model for Multi-tab Website Fingerprinting Attack. In: CCS (2023)
35. Kadloor, S., Gong, X., Tezcan, T., Borisov, N.: Low-Cost Side Channel Remote Traffic Analysis Attack in Packet Networks. In: IEEE ICC (2010)
36. Khan, M.U.S., Bukhari, S.M.A.H., Maqsood, T., Fayyaz, M.A.B., Dancey, D., Nawaz, R.: SCNN-Attack: A Side-Channel Attack to Identify YouTube Videos in a VPN and Non-VPN Network Traffic. Electronics **11**(3) (1 2022)
37. Khan, M.U., Bukhari, S.M., Khan, S.A., Maqsood, T.: ISP can identify YouTube videos that you just watched. In: IEEE FIT (2021)
38. Kirchner, R., Koch, S., Kamangar, N., Klein, D., Johns, M.: A Black-Box Privacy Analysis of Messaging Service Providers' Chat Message Processing. Privacy Enhancing Technologies (2024)
39. Korczyński, M., Duda, A.: Markov chain fingerprinting to classify encrypted traffic. In: IEEE Conference on Computer Communications (2014)
40. Lastovicka, M., Jirsik, T., Celeda, P., Spacek, S., Filakovsky, D.: Passive OS Fingerprinting Methods in the Jungle of Wireless Networks. In: NOMS (2018)
41. Lau, F., Rubin, S.H., Smith, M.H., Trajkovic, L.: Distributed Denial of Service Attacks. In: International Conference on Systems, Man and Cybernetics (2000)
42. Lescisin, M., Mahmoud, Q.: Tools for Active and Passive Network Side-Channel Detection for Web Applications. In: WOOT (2018)
43. Liang, B., You, W., Liu, L., Shi, W., Heiderich, M.: Scriptless Timing Attacks on Web Browser Privacy. In: Annual IEEE/IFIP International Conference on Dependable Systems and Networks (2014)
44. Luckie, M., Beverly, R., Brinkmeyer, W., Claffy, K.: Speedtrap: Internet-Scale IPv6 Alias Resolution. In: IMC (2013)
45. Lyon, G.: Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning. Insecure (2009)

46. Mittal, P., Khurshid, A., Juen, J., Caesar, M., Borisov, N.: Stealthy traffic analysis of low-latency anonymous communication using throughput fingerprinting. In: CCS (2011)
47. Msadek, N., Soua, R., Engel, T.: IoT device fingerprinting: Machine learning based encrypted traffic analysis. In: Wireless Communications and Networking Conference (WCNC) (2019)
48. Murdoch, S.J., Danezis, G.: Low-cost traffic analysis of Tor. In: S&P (2005)
49. Open Graph: The Open Graph protocol (2025), `https://ogp.me/`
50. Panchenko, A., Lanze, F., Pennekamp, J., Engel, T., Zinnen, A., Henze, M., Wehrle, K.: Website Fingerprinting at Internet Scale. In: NDSS (2016)
51. Panchenko, A., Niessen, L., Zinnen, A., Engel, T.: Website Fingerprinting in Onion Routing Based Anonymization Networks. In: WPES (2011)
52. Rahman, M.S., Mathews, N., Wright, M.: Video Fingerprinting in Tor. In: CCS (2019)
53. Rauscher, F., Kogler, A., Juffinger, J., Gruss, D.: IdleLeak: Exploiting Idle State Side Effects for Information Leakage. In: NDSS (2024)
54. Reed, A., Kranch, M.: Identifying HTTPS-Protected Netflix Videos in Real-Time. In: CODASPY (2017)
55. Reed, M., Syverson, P., Goldschlag, D.: Anonymous Connections and Onion Routing. Journal on Selected Areas in Communications **16**(4), 482–494 (1998)
56. Rimmer, V., Preuveneers, D., Juarez, M., Van Goethem, T., Joosen, W.: Automated website fingerprinting through deep learning. In: NDSS (2017)
57. Schuster, R., Shmatikov, V., Tromer, E.: Beauty and the Burst: Remote Identification of Encrypted Video Streams. In: USENIX Security (2017)
58. Shen, M., Gao, Z., Zhu, L., Xu, K.: Efficient fine-grained website fingerprinting via encrypted traffic analysis with deep learning. In: International Symposium on Quality of Service (IWQOS) (2021)
59. Shen, M., Ji, K., Gao, Z., Li, Q., Zhu, L., Xu, K.: Subverting Website Fingerprinting Defenses with Robust Traffic Representation. In: USENIX Security (2023)
60. Shen, M., Liu, Y., Zhu, L., Du, X., Hu, J.: Fine-grained webpage fingerprinting using only packet length information of encrypted traffic. TIFS **16**, 2046–2059 (2020)
61. Shen, M., Wei, M., Zhu, L., Wang, M.: Classification of encrypted traffic with second-order markov chains and application attribute bigrams. TIFS **12**(8), 1830–1843 (2017)
62. Shintre, S., Gligor, V., Barros, J.: Optimal strategies for side-channel leakage in FCFS packet schedulers. In: International Symposium on Information Theory (ISIT) (2015)
63. Singh, A., Nordström, O., Lu, C., Dos Santos, A.L.: Malicious ICMP tunneling: Defense against the vulnerability. In: Australasian Conference on Information Security and Privacy (ACISP) (2003)
64. Sirinam, P., Imani, M., Juarez, M., Wright, M.: Deep Fingerprinting: Undermining Website Fingerprinting Defenses with Deep Learning. In: CCS (2018)
65. Sirinam, P., Mathews, N., Rahman, M., Wright, M.: Triplet Fingerprinting: More Practical and Portable Website Fingerprinting with N-shot Learning. In: CCS (2019)
66. Skowron, M., Janicki, A., Mazurczyk, W.: Traffic fingerprinting attacks on internet of things using machine learning. IEEE Access **8**, 20386–20400 (2020)
67. Song, D.X., Wagner, D., Tian, X.: Timing Analysis of Keystrokes and Timing Attacks on SSH. In: USENIX Security (2001)

68. Spring, N., Mahajan, R., Wetherall, D.: Measuring ISP Topologies with Rocketfuel. In: SIGCOMM (2002)
69. Stivala, G., Pellegrino, G.: Deceptive Previews: A Study of the Link Preview Trustworthiness in Social Platforms. In: NDSS (2020)
70. Trampert, L., Schwarz, M.: Hidden in Plain Sight: Scriptless Microarchitectural Attacks via TrueType Font Hinting. In: uASC (2025)
71. Trampert, L., Weber, D., Gerlach, L., Rossow, C., Schwarz, M.: Cascading Spy Sheets: Exploiting the Complexity of Modern CSS for Email and Browser Fingerprinting. In: NDSS (2025)
72. Walsh, T., Thomas, T., Barton, A.: Exploring the Capabilities and Limitations of Video Stream Fingerprinting. In: S&P Workshops (2024)
73. Wang, T., Cai, X., Nithyanand, R., Johnson, R., Goldberg, I.: Effective Attacks and Provable Defenses for Website Fingerprinting. In: USENIX Security (2014)
74. Wang, T., Goldberg, I.: Improved Website Fingerprinting on Tor. In: WPES (2013)
75. Wang, X., Luo, J., Yang, M., Ling, Z.: A potential HTTP-based application-level attack against Tor. Future Generation Computer Systems (2011)
76. White, A., Matthews, A., Snow, K., Monrose, F.: Phonotactic Reconstruction of Encrypted VoIP Conversations: Hookt on Fon-iks. In: S&P (2011)
77. Wright, C., Bellard, L., Monrose, F., Masson, G.: Language Identification of Encrypted VoIP Traffic: Alejandra y Roberto or Alice and Bob? In: USENIX Security (2007)
78. Xue, D., Kallitsis, M., Houmansadr, A., Ensafi, R.: Fingerprinting Obfuscated Proxy Traffic with Encapsulated TLS Handshakes. In: USENIX Security (2024)
79. Zalewski, M.: p0f v3 (3.09b) (2014), `https://lcamtuf.coredump.cx/p0f3/`
80. Zhu, Y., Graham, B., Bettati, R., Zhao, W.: Correlation-Based Traffic Analysis Attacks on Anonymity Networks. IEEE Transactions on Parallel and Distributed Systems (2010)