In Proceedings of 14th USENIX Workshop on Offensive Technologies (WOOT'20)

Bankrupt Covert Channel: Turning Network Predictability into Vulnerability

Dmitrii Ustiugov[*] Plamen Petrov[*] M.R. Siavash Katebzadeh Boris Grot

University of Edinburgh

Abstract

Recent years have seen a surge in the number of data leaks despite aggressive information-containment measures deployed by cloud providers. When attackers acquire sensitive data in a secure cloud environment, covert communication channels are a key tool to exfiltrate the data to the outside world. While the bulk of prior work focused on covert channels within a single CPU, they require the spy (transmitter) and the receiver to share the CPU, which might be difficult to achieve in a cloud environment with hundreds or thousands of machines.

This work presents Bankrupt, a high-rate highly clandestine channel that enables covert communication between the spy and the receiver running on different nodes in an RDMA network. In Bankrupt, the spy communicates with the receiver by issuing RDMA network packets to a private memory region allocated to it on a different machine (an intermediary). The receiver similarly allocates a separate memory region on the same intermediary, also accessed via RDMA. By steering RDMA packets to a specific set of remote memory addresses, the spy causes deep queuing at one memory bank, which is the finest addressable internal unit of main memory. This exposes a timing channel that the receiver can listen on by issuing probe packets to addresses mapped to the same bank but in its own private memory region. Bankrupt channel delivers 74Kb/s throughput in CloudLab's public cloud while remaining undetectable to the existing monitoring capabilities, such as CPU and NIC performance counters.

1 Introduction

In the digital era, information security has become the crucial necessity that drives private and public cloud vendors to take all available measures to contain sensitive and confidential data within customer and government-defined boundaries. Despite the efforts, security breaches are commonplace, often compromising highly sensitive data including personal information [15, 23, 50, 52], passwords [11, 16, 57], and medical records [19, 20].

Often staying in the shadow of side channels that acquire the sensitive data, covert communication channels are a critical tool used by attackers to exfiltrate the data from a secure environment. Due to strict information containment measures, like firewalls, a *spy* software may not have a direct access to the Internet and, instead, has to communicate – via a covert channel – the acquired data to a co-operative *receiver* software that is outside of the secure environment and with Internet access. Recent study by Reardon et al. spotlights a wide usage of covert channels by attackers in the real world [46].

From an attacker's perspective, constructing an efficient covert channel poses several practical challenges. In a public or private cloud with thousands of nodes, it may be difficult to ensure that spy and receiver applications get scheduled to the same node. A more reliable strategy is to construct a covert channel that works across the datacenter, thus allowing the spy and the receiver to reside on different nodes. Second, the channel needs to be fast enough to transmit any amount of sensitive data, which could reach into gigabytes (e.g., medical records [19, 20]). Finally, the channel must remain stealthy even if the cloud vendor is actively monitoring for its existence.

In this work, we introduce a timing-based covert communication channel, called Bankrupt, that meets all of these requirements. Bankrupt enables high-rate covert communication across a datacenter's RDMA-enabled network, thus decoupling the physical placement of the spy and the receiver from their ability to communicate covertly while avoiding detection by existing monitoring facilities, including CPU and NIC performance counters.

Bankrupt relies on an RDMA-enabled network, a technology that is being adopted by public and private cloud operators [4, 18, 22, 38]. The wide deployment of RDMA technology is due to its fundamental latency advantages over traditional datacenter networks. These advantages come from the fact that RDMA offers direct access to a remote server's memory via an RDMA-enabled NIC with full bypass of the remote CPU. The Bankrupt attack turns the low latency and predictability of RDMA into a vulnerability, as predictable latencies enable a highly robust timing channel.

Bankrupt uses a remote node's main memory as a timing channel accessed via RDMA. Figure 1 shows how the attack works. In the figure, the spy (also referred to as a sender) wishes to communicate a secret to the receiver residing on a

[*] The first two authors contributed equally to this work.

different node. Although unable to communicate directly, e.g., in case of logical separation of the network, both the sender and the receiver have access to another node's (an intermediary) memory (e.g., a storage server [37, 47]), where both the sender and the receiver have legitimately allocated private RDMA-exposed memory regions. The sender and the receiver do *not* share physical memory, thus being isolated from the cloud operator's perspective. The sender sends a stream of RDMA read requests to its private memory region at the intermediary, but concentrates all of the reads on a single memory bank inside the intermediary's memory.[1] By exploiting the much higher bandwidth of RDMA (up to 200Gb/s with today's commodity NICs) in comparison to that of a single memory bank (~10Gb/s), the spy can trivially induce queuing at the target bank, causing the latency of accesses to that bank to spike. By issuing RDMA reads to the same bank, the receiver can detect the latency spike. The presence or absence of high latency, as in a modulated analog signal, informs the receiver of the current bit's value.

In order for Bankrupt to succeed, both the spy and the receiver must be able to consistently target a single memory bank on a server with hundreds of banks. A memory controller inside the intermediary CPU uses a subset of physical address bits (the bank bits) to route the memory requests, originated both from the CPU and the RDMA NIC, to the target bank. As RDMA legitimately exposes the virtual addresses of the intermediary, we show that a remote attacker is able to use this information to reverse-engineer the position of the bank bits in the address space of the intermediary with an algorithm that is of linear complexity in the number of address bits.

Another challenge for Bankrupt is to guarantee that the latency spike on the target node is sufficiently high that the receiver on another node can reliably observe it by issuing probes over RDMA to the same bank. The quality of the signal may be compromised by network noise or the presence of memory-intensive applications on the intermediary. However, our experiments show that the massive bandwidth offered by today's RDMA networks and memory subsystems create a favorable environment for a highly robust channel.

We evaluate the Bankrupt channel in a private cluster and on CloudLab, a large-scale public cloud used by computer researchers. In the private cluster, the Bankrupt channel achieves 114Kb/s throughput in isolation. Under network load, Bankrupt still provides 67Kb/s throughput, as the sender needs to issue larger bursts of RDMA reads to overcome the noise level. We find that the channel's bandwidth and accuracy are unaffected in the presence of local memory traffic from innocuous memory-intensive applications that concurrently run on the intermediary because a CPU efficiently balances regular memory traffic. On CloudLab, we show that Bankrupt delivers 74Kb/s channel throughput.

To summarize, our contributions are as follows:

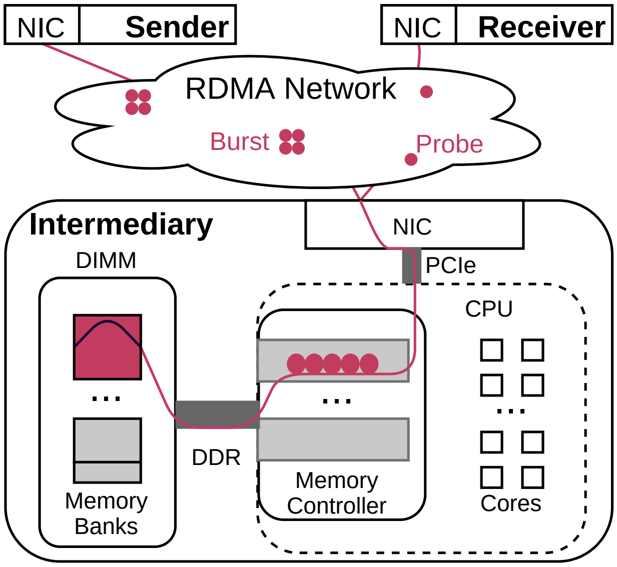


Figure 1: Bankrupt attack environment and operation.

- We construct a timing channel, called Bankrupt, that enables covert communication between participants, which share the same RDMA network but run on different nodes, via queuing inside remote memory hosted on a third – innocuous – node.
- We provide guidelines for setting up Bankrupt with an arbitrary RDMA network and CPU; namely for discovering the addresses that belong to the target memory bank in remote memory as well as establishing and dynamically adjusting the channel's modulated signal by issuing shaped bursts of RDMA traffic even in a noisy environment.
- We show that the attack is undetectable by software-based memory access latency monitoring, Infiniband NIC and CPU performance counters.

2 Background

The Bankrupt channel leverages features of a modern RDMA-enabled network and the CPU memory subsystem. This section introduces the relevant aspects of these technologies in order to explain how they work together in Bankrupt.

2.1 Covert Channels

Covert channels, introduced first by Lampson [28], enable communication between independent entities over isolation boundaries, bypassing firewalls and communication auditing. The basic setup of a covert channel consists of the two participants: a spy (or sender) program that would like to communicate (exfiltrate) the previously obtained secret to the outside world, and a receiver that is aware of the sender's existence. Finally, the sender and the receiver should share

[1] A memory bank is a memory-internal device that hosts the data. A modern memory subsystem comprises up to hundreds of banks (§2.3).

a common resource that they can use to communicate and decode the signal.

To communicate, the sender and the receiver usually exploit resource sharing in a computer system by establishing a communication channel via legitimate innocuous actions. For example, a timing channel can be established by modulating access latencies to a shared CPU cache [29, 31, 56], main memory [43, 54, 55] or by modulating temperature on a multi-core CPU chip [30].

Despite the variety of the previously proposed covert channels, only a few of them may pose a threat in a realistic deployment scenario, as indicated by prior work [53]. We identify three key requirements that a practical covert channel has to meet for maximum impact and generality, namely *wide visibility*, *high communication bandwidth*, and *stealthiness*. We next discuss each of these requirements.

Wide visibility: An attacker seeking to exfiltrate secrets needs a covert channel that avoids or minimizes the degree of colocation between the sender and the receiver. While the most straight-forward way for the sender and the receiver to covertly communicate is by co-running on the same node, such a high degree of colocation is difficult to achieve in practice because cloud vendors purposefully randomize placement of the applications on their nodes. However, many providers do provide their customers with means to localize applications within the same physical network to offer low latency networking [7, 8]. Hence, a cross-network covert channel may significantly simplify the attack setup for an attacker.

High bandwidth: Certain types of sensitive data may have a significant volume, potentially reaching into gigabytes (e.g., healthcare records [20]). Hence, to be broadly applicable, a covert channel should deliver high transmission bandwidth channel to exfiltrate data of an arbitrary size. Moreover, a channel's bandwidth should remain high even in the presence of various types of system noise, e.g., network traffic or CPU activity of numerous innocuous applications and system services that share the same network or CPU.

Stealthiness: An ideal covert channel should remain stealthy even if platform owners suspect its existence and actively monitor for it. Indeed, modern servers feature a wide range of monitoring capabilities, for example, CPU performance counters and network latency anomaly detectors [3, 10, 12, 26, 44, 45, 58]. A truly covert channel should remain undetectable by these techniques.

Previously proposed covert channels struggle to comply with all three requirements. CPU-cache based covert channels restrict the sender and receiver colocation scenario to the same physical core or CPU chip [29, 31, 53, 56]. Cache-based channels also tend to increase the miss rate of the caches exposing themselves to system monitoring software [17, 53]. The attacks across a network often struggle to deliver bandwidth higher than 1Kb/s [41, 48] or may be difficult to use in a noisy environment due to a relatively small latency gap in their timing channel [27].

2.2 RDMA Networks

Originating from high-performance computing, Remote Direct Memory Access (RDMA) networks see rapid adoption by cloud providers that strive to deliver low latency and high throughput to their customers at the scale of an entire datacenter [4, 18, 22, 38]. For example, Azure customers can rent a virtual RDMA-connected cluster of 80 000 virtual cores [39].

RDMA allows the applications to access the remote memory via user-level *one-sided* RDMA read and write primitives that bypass the destination CPU completely, allowing network packets to directly read or write to its memory. First, to expose remote memory for RDMA operations, an application running on one node needs to register a memory region on the node that hosts the memory. After that, the application can issue RDMA read (write) network packets to read (write) the remote memory contents directly – by specifying the remote node's virtual address and the accessed memory chunk length.

When an RDMA packet arrives at the destination node's NIC, the NIC translates the virtual address, specified in the packet's header, to the corresponding physical memory address and engages a DMA (read or write) transaction. The root complex unrolls the transaction into a sequence of CPU cache-block sized memory requests that eventually reach a memory controller of the CPU. Finally, the memory controller serves these requests from the destination's main memory and sends the responses back to the NIC. Upon receiving the responses from the memory controller, the NIC forms a response packet and sends it out to the requesting node.

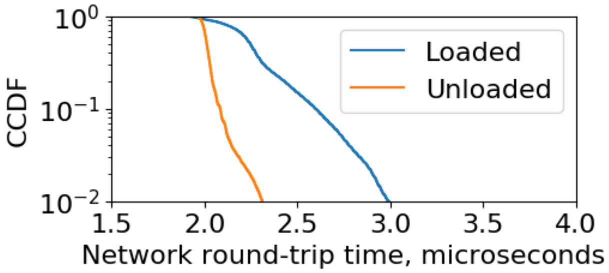


Figure 2: Round-trip latency, as a complementary cumulative distribution function (CCDF), of an RDMA read operation in isolation and when network links and switch run at 70% utilization. See §5 for the setup details.

Because RDMA bypasses the destination CPU, the network round-trip time remains predictable even in the presence of other network-intensive applications that use the network infrastructure simultaneously. To showcase RDMA network predictability, we plot the 64B RDMA read latency, in isolation and under network load of ~40Gb/s (which represents ~70% of the link bandwidth) from a client node to a server node (Figure 2). Despite the loaded network links and switch ports, the 99^{th} percentile of the network round-trip delay is larger than in the unloaded case by just 1.2 microseconds. This high RDMA predictability indicates the potential for constructing a robust timing channel in an RDMA network.

2.3 Memory Organization

The memory subsystem of a modern server CPU has a hierarchical structure: there are 2-6 memory channels, each with its own memory controller, and each channel has 2-4 DIMMs. Each DIMM consists of up to 64 independent memory devices called memory banks.[2] In total, today's server may have hundreds of memory banks across its multitude of DIMMs, channels and sockets [5].

The high overall bandwidth of a modern memory system is delivered by this ensemble of small independent banks, where each bank provides only a small fraction of the total memory bandwidth. Internally, a memory bank stores data as an array of DRAM rows, bringing one row at a time to an SRAM row buffer before serving a memory request. A single bank's bandwidth is bounded by the time the bank takes to bring and serve the corresponding DRAM row from the array. Assuming no access locality, this time is the sum of the three key DRAM latency components, namely *tCL*, *tRCD*, and *tRP*. Each of the three latency components is 13-15ns which means that the total latency of serving a single 64-byte memory request is 39-45ns, resulting in the peak theoretical 1.4-1.6GB/s bandwidth of a single bank. Due to the end of DRAM technology scaling, these key latency components have stayed the same over the last two decades [35, 36].

The memory controller manages each bank independently from others, hence all requests to a particular bank reside in a dedicated per-bank queue inside the memory controller. The memory controller routes memory requests to banks based on their physical addresses by employing a manufacturer-specific fixed hash function (referred to as a memory interleaving scheme) that determines the destination bank based on a subset of the memory request's physical address.

To deliver maximum overall memory bandwidth, manufacturers choose the scheme that maximises the parallelism across memory banks. For example, many processors use a cache-block interleaving scheme that implies that accesses to adjacent cache blocks are served from different banks. While some manufacturers disclose memory interleaving information, others keep this information proprietary [6, 43].

3 Threat Model

We assume a highly restricted setting that is similar to the public cloud environment [4, 18, 22, 38]. Both sender and receiver run either natively or in a virtual machine. They do not have means to guarantee colocation on the same server, and thus, might be unable to leverage one of the known local (i.e., on the same node) covert channels. Furthermore, the sender and receiver are not allowed to directly communicate with each other over the network, as if the network is logically separated. We assume that no privilege escalation is possible and both the sender's and the receiver's applications have normal user privileges. The applications are unable to alter any of the existing firewall policies and do not have the capability to observe the network traffic.

To establish a covert communication channel in this environment, the sender and the receiver can instead communicate via an intra-datacenter network by modulating local or remote memory access latency. For example, both the sender and the receiver may be able to allocate non-shared remote memory regions on one of the shared storage servers [37, 47], as assumed by prior work [27]. The remote memory access latency can be modulated by performing one-sided read and/or write operations via RDMA. Ideally, the sender can allocate remote memory on a receiver's node, allowing the receiver to observe memory access latency on its own server. However, this might be difficult to guarantee as the sender has no knowledge where the receiver is running, and furthermore, is unable to control where its remote memory is allocated by the datacenter resource manager.

With Bankrupt, the sender and the receiver can communicate by using the remote memory of one of the other nodes in the cluster (further referred to as an intermediary), where both sides can allocate remote memory. To find an intermediary, both the sender and the receiver may request many storage servers and periodically create and search for the modulated signal inside the remote memory of each server by periodically probing a specific set of addresses (§4.2.1) in each server's memory with RDMA reads.

The intermediary is non-malicious and the attacker does not have direct control over it. The sole requirement is that the sender and the receiver need to have access to their corresponding – completely private – memory regions that have been allocated for their exclusive use with RDMA one-sided operations (i.e., remote reads and writes).

4 The Bankrupt Channel

The Bankrupt attack turns the main advantage of RDMA networks – the direct access to remote memory – into a high-rate stealthy communication channel that remains robust even in the presence of noise arising from the network and other innocuous co-running applications/VMs.

In this section, we describe the attack, introducing a cross-network timing channel, and provide guidelines for setting up the Bankrupt channel in an arbitrary RDMA cluster.

4.1 The Bankrupt Timing Channel

Direct access to remote memory enables the sender to create a fast timing channel by modulating the access latency to a single memory bank by selectively steering legitimate network traffic to a small subset of memory addresses in the intermediary's memory.

[2] DIMMs normally consist of 1-4 ranks, and each rank of 8-16 banks.

The timing channel relies on several features of the RDMA NIC and the CPU in a modern server. First, CPU and RDMA network cards are tightly integrated via PCIe. This allows the network traffic to flow directly into the destination CPU's memory subsystem without any software mediation from the destination CPU (§2.2). Second, the memory bank addressing (memory interleaving) scheme is static, which enables a remote attacker to steer their network traffic to a single memory bank (§2.3). Finally, the bandwidth disparity between the 100-200Gb/s network and ~10Gb/s memory bank enables a remote attacker to easily modulate the access latency of a single memory bank by forcing queuing at the target memory bank with excessive traffic.

To set up reliable communication, we use a unidirectional communication protocol. The sender transmits information as a synchronous analog signal by modulating, at a fixed frequency, the response latency of a single bank that is located in remote memory. By doing that, the sender modulates the entire network round-trip delay, as observed by the receiver, for those RDMA packets that go to the target bank in the remote memory. To transmit a `1`, the sender switches the target memory bank to the *contended* state (i.e., high access time during a period) by steering the network traffic in the form of bursts of RDMA read operations to the bank.[3] To transmit a `0`, the sender does not issue network traffic, leaving the bank in the *uncontended* state (i.e., low access time) for the time period that is defined by the sending frequency.

To read the transmitted signal, the receiver issues probes, which are normal RDMA read operations, at a certain frequency to measure and record the observed remote memory access latency. The sender splinters the messages into fixed-sized packets with a pre-defined preamble as a packet header, followed by a payload. The preamble contains a number of bits that would indicate the beginning of a message to the receiver while the payload contains the transmitted information. Transmission accuracy can be further improved by adding error correction codes to the packet header, though we do not take advantage of it in this work.

4.2 Bankrupt Setup Guidelines

To set up the channel, both the sender and the receiver need to find a subset of remote memory addresses that belong to the same memory bank in the intermediary's memory – this bank will serve as the actual timing channel. The sender needs to adjust the size of the bursts of RDMA operations and to select the appropriate sending frequency. The receiver needs to tune its probing frequency. We next discuss each of these requirements.

4.2.1 Identifying RDMA Addresses that Belong to the Same Memory Bank

The sender and the receiver (which, together, we refer to as the "attacker") can use an arbitrary bank in the intermediary. The choice of the bank is defined by the subset of the virtual address bits (further referred to as *bank bits*), the value of which tells the memory controller which bank to route a memory request to (§2.3). For simplicity, the attacker can set the bank bits to a pre-defined value (e.g., all zeros). The attacker (both the sender and the receiver can do it separately) just needs to find any addresses with bank bits set to that value.

The attacker starts by locating the bank bits. These bits are normally located in the least-significant, i.e., [6:X] where $X > 6$, part of the address in order to balance the load across all available banks. Hence, the attacker needs to derive X to locate the positions of the bank bits.

Since modern computer systems manage memory at the granularity of pages (Linux/x86 uses 4KB, 2MB, and 1GB pages), a subset of least-significant bits (LSBs) in a virtual address and the corresponding physical address are identical. In RDMA-connected systems where low latency is a priority, the vendors often use large, 2MB or 1GB, pages for remote memory to avoid the bulk of translation cache misses in RDMA NICs [13, 40, 51, 59]. If 2MB or 1GB pages are used, virtual and physical addresses share 21 or 30 LSBs, respectively. Knowing 30 bits is enough to locate the positions of all the bits in a virtual address that identify a bank for all the tested systems both in this work and in prior work [43]. For some systems, knowing 21 LSB bits suffices.

The attacker can identify the positions of the bank bits in an address by using only RDMA read operations in a search algorithm, which is of linear complexity in the number of address bits. This algorithm allows to find addresses that belong to a single bank by iteratively excluding addresses belonging to other banks.

First, the attacker chooses a set of unique arbitrary virtual addresses that are 64-byte (cache-block) aligned, i.e., the bits [0:5] in the address must be zero while bits [6:63] have arbitrary values. The number of addresses needs to be big enough to not fit in a memory controller's hardware buffer at once to avoid coalescing requests to the same address. We find that 64 addresses are enough for all the platforms that we tested.

Second, the attacker issues RDMA reads to those addresses and estimates the resulting network throughput based on the number of RDMA reads that complete in the measurement interval. If the traffic exceeds the throughput of a single memory bank (~1.2GB/s in our experiments), the attacker infers that memory requests are being served by several banks. To exclude addresses from a half of the banks in each iteration, the attacker sets one more LSB bit to 0 ([0:6] bits in the second iteration, [0:7] in the third one, etc.) and estimates the network throughput. After a number of iterations, the throughput converges to that of a single bank, meaning that the most-

[3] Cache blocks, accessed by RDMA reads, are not allocated in a CPU's last-level cache to avoid cache pollution [24].

significant bank bit is set to zero, revealing the X parameter. Note, that not all bits in [6:X] define a bank so some iterations may not lead to a throughput reduction.

4.2.2 Tuning Burst Characteristics

The sender modulates the access time of the target memory bank by forcing queue build-up at the target bank's queue inside the memory controller. To force queuing, the sender issues bursts of RDMA reads to addresses that map to the target bank. Below we provide guidelines for the sender to shape the traffic to produce a robust signal that is visible across the RDMA network.

To guarantee visibility of the signal, the sender must issue bursts that require a large amount of time to drain by the target bank (e.g., a microsecond or more). A memory bank serves read accesses to any of these addresses serially and each one within a fixed service time that is in the 39-45ns range (§2.3). This enables the sender to accurately estimate the service time of a bank depending on the queue depth. Given the predictability of the RDMA network (§2.2), it is possible to accurately modulate the RDMA network round-trip delay.

4.2.3 Tuning Sender's and Receiver's Frequency

To maximize the transmission bandwidth of the channel, the sender needs to find the maximum frequency, i.e., the minimum period, which depends on the time the target bank takes to serve a burst as well as the level of noise in the system. There are two types of noise that may impact Bankrupt's transmission speed and accuracy. First is the network traffic from other innocuous applications, which share the same physical network, that may increase network delays. The other type of noise is the memory traffic that comes from innocuous software that runs on the intermediary, causing extra queuing at the memory banks.

To overcome the noise, the sender may need to increase the burst size so that the modulated delay is higher than the noise level, i.e., maintain the signal/noise ratio more than 1. Hence, the sender should periodically adjust its frequency according to the environment changes: larger bursts improve the signal/noise ratio but take longer to drain which limits the sending frequency. To determine the optimal sending frequency, both the sender and the receiver measure the current network round-trip time (which serves as an indicator of the noise level) and determine the minimal required burst size for the current noise level. Given the pre-defined preamble in packet headers, the receiver can derive the sending frequency and adjust to its changes.

The receiver needs to issue probes, in the form of RDMA reads, to the intermediary's bank to detect the transmitted signal. The probing frequency needs to be high enough to allow the receiver to decode the bit transmitted in each sending period. Before the transmission starts, the receiver first determines the unloaded latency as the 95^{th} percentile latency of the network round-trip when the channel is inactive, to discount the rare RDMA network latency outliers. Then, during each sending period, the receiver performs several measurements of round-trip time and compares the round-trip latency to the unloaded latency to determine if the target bank is in the contended state. The receiver computes the round-trip latency as the arithmetic mean of the measurements that are taken in the middle of that period to capture the peak of the potential bank contention, by dropping first and last 25% of the measurements. If the round-trip latency is larger than the unloaded latency, the receiver records `1`, otherwise a `0`. To account for sporadic changes in network load, the receiver periodically re-computes the unloaded latency.

	Private Cluster	**CloudLab Utah**
CPU	Xeon E5-2630v4 @2.20GHz (Broadwell)	Xeon E5-2640v4 @2.40GHz (Broadwell)
RAM	4×16GB, DDR4-2400	4×16GB, DDR4-2400
NIC	Mellanox MT27800 CX-5, 56Gb/s	Mellanox MT27710 CX-4 Lx, 50Gb/s
Core switches	—	1 Mellanox 2700
ToR switches	Mellanox SX6012	5 Mellanox 2410
OS	Ubuntu 18.04	Ubuntu 16.04
Kernel	4.15.0-58-generic	4.15.0-88-generic
Nodes	6	200

Table 1: Specifications of studied platforms.

5 Methodology

We evaluate Bankrupt in terms of transmission bandwidth, accuracy, and stealthiness.

We mount Bankrupt on a private cluster to examine the performance in isolation and under various types of load. To demonstrate the feasibility of the Bankrupt channel in a public cloud, we evaluate it on CloudLab [2, 14] – a public cloud in use by Computer Science researchers. We evaluated the Bankrupt channel using CloudLab's Utah cluster that connects 200 nodes with a high-speed RDMA network. At the time of testing, the utilization of the nodes in the cluster was 80%. The specifications of our private cluster and the CloudLab cluster are presented in Table 1. Similarly to prior work [43], we enable 1GB large pages on our experimental platforms as they are widely used in RDMA deployments [40, 51, 59]. In all of our experiments, the sender and the receiver use RDMA reads of 64 Bytes.

To evaluate the bandwidth and accuracy of transmission over the Bankrupt channel, we send a sequence of packets where each packet carries a constant preamble and a randomly-generated payload. For all experiments, unless stated otherwise, we use a 32-bit long preamble of `10..10` to tune the decoder before decoding the payload of 200 bits. We report channel throughput as its *true channel capacity*, which we define as achieved channel bandwidth discounting preambles and incorrectly transmitted bits. We measure accuracy as the fraction of correctly decoded bits in the message that we are sending. We keep the receiver's probing frequency set to 500ns in all our experiments as we found that its impact on accuracy is rather moderate.

6 Evaluation

In this section we demonstrate the properties of the Bankrupt attack in isolation and under various types of load. We conclude the section with an evaluation of the stealthiness characteristics of the channel and its performance in a realistic datacenter environment.

6.1 Performance in Isolation

Following the algorithm in §4.2.1, we found that the bank bits are located at the positions of bits [6:26] in the address.

Figure 3 shows that the latency measurements during the transmission for different burst sizes. Note that each subfigure shows a different number of transmitted bits in a fixed amount of time (200 microseconds) because varying the burst size also changes the (maximum) sending frequency (§4.2.3). The difference between latencies, further referred to as the *latency gap*, that are measured during transmitting `1`-s and `0`-s grows with the burst size. The minimal burst size that allows decoding is 32 (2KB) that shows the latency gap of 0.5 microsecond. With the burst size of 128 (8KB), the latency gap is more than twice bigger, around a microsecond that is comparable to the network round-trip time, making the signal more pronounced.

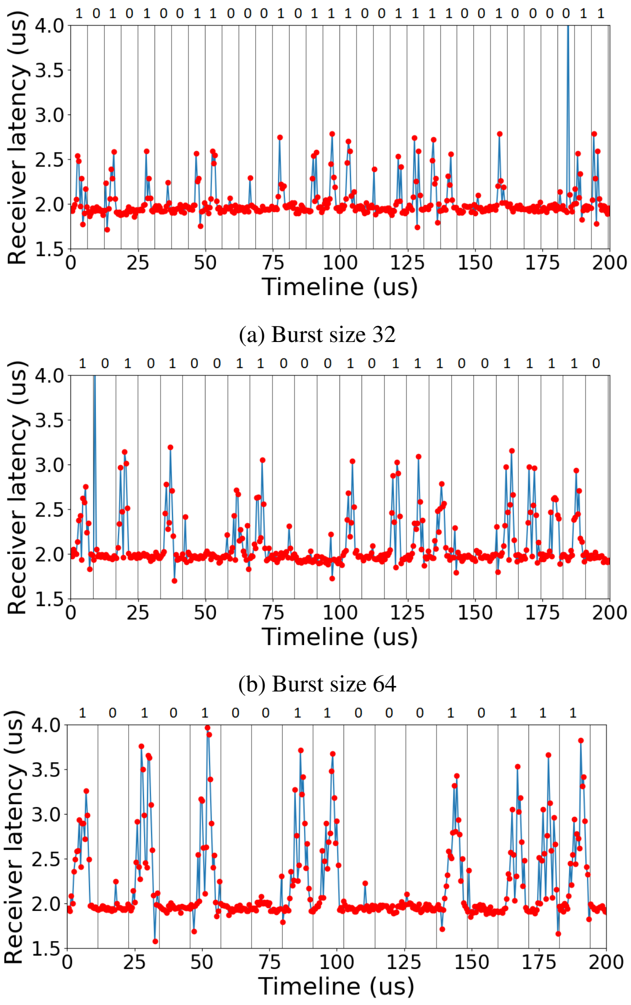
Figure 4 shows how the channel throughput and transmission accuracy depend on the burst size. Increasing the burst size from 16 to 32 nearly doubles the transmission accuracy that leads to a surge in the channel throughput, despite the lower signaling frequency with a larger burst size. However, increasing the burst size beyond 32 results in a diminished channel throughput as large burst sizes decrease the signaling frequency that is not compensated for by the gain in accuracy. Thus, we find that the burst size of 32 delivers the highest channel throughput of 114Kb/s (with transmission accuracy of 82.4%) in isolation in the private cluster.

6.2 Performance Under Load

To evaluate the channel in the presence of noise, we model both the noise coming from other workloads scheduled on the intermediary's CPU and network traffic.

6.2.1 Local Load

To model the local load in the intermediary's memory, we launch 16 instances of the `mcf` benchmark, taken from the popular SPEC 2006 benchmark suite [21], on the intermediary node. `mcf` is the most memory-intensive benchmark from that suite [25]. 16 instances of `mcf` that together generate a variable load on the intermediary's memory ranging from 2GB/s to 8GB/s. This is equal to 32MB/s to 128MB/s for each bank, assuming that on average the load spreads evenly among banks.



(a) Burst size 32

(b) Burst size 64

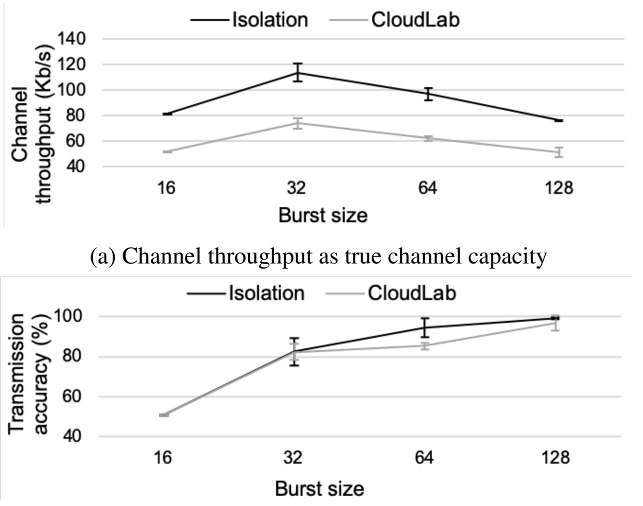
(c) Burst size 128

Figure 3: Signal observed by the receiver in isolation in the private cluster, with the corresponding transmitted messages.

Figure 5 shows the signal observed by the receiver for the burst size of 32 in the presence of the local load. The signal is indistinguishable from the one observed in isolation for the same burst size (Figure 3). The throughput and the accuracy remain the same as in isolated execution. The channel is unaffected by other workloads running on the intermediary because each individual bank receives a relatively small amount of traffic, since the load is spread across all of the banks in the system (§2.3).

6.2.2 Network Load

To model network load on the intermediary, we use the `ib_read_bw` benchmark, which is part of the RDMA Perftest



(a) Channel throughput as true channel capacity

(b) Transmission accuracy

Figure 4: True channel capacity and transmission accuracy of the Bankrupt covert channel in isolation and in CloudLab. Error bars show the standard deviation.

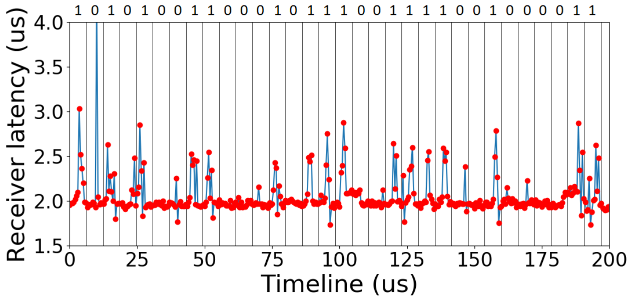
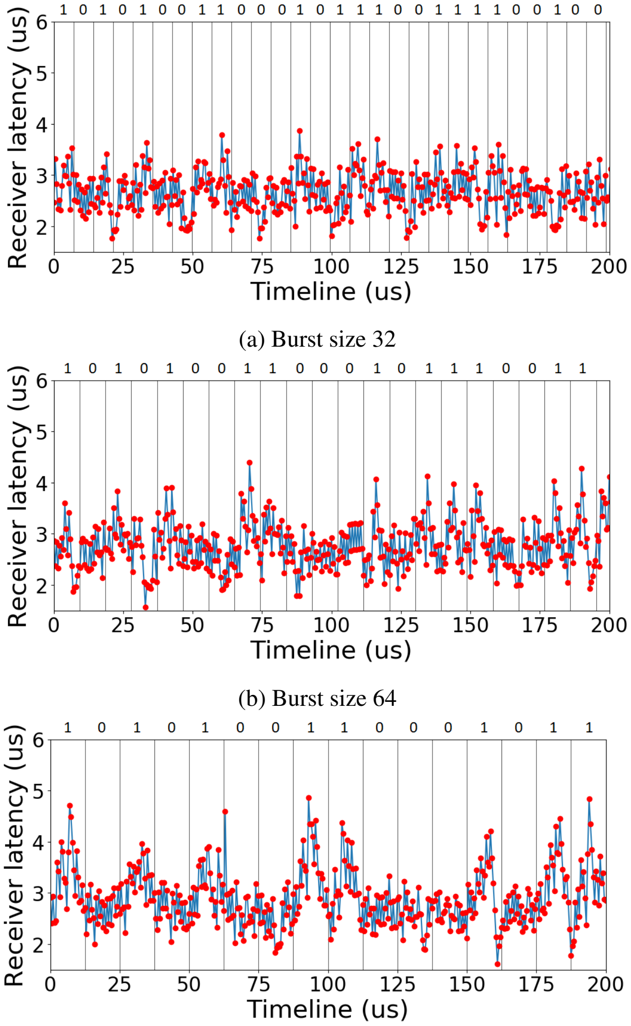


Figure 5: Signal observed by the receiver in the presence of local load in the private cluster. The burst size is 32.

[34]. We launch ib_read_bw from a node, which is different from the three nodes we use for the channel. The network loader generates network traffic to the intermediary, loading the network link, top-of-the-rack switch, and the NIC of the intermediary. The generated load is equal to ~40Gb/s which is around 70% of the intermediary's total link bandwidth.

Figure 6 shows the signal observed by the receiver in the presence of network load. With the burst size of 32, the signal is noisy, which prevents its decoding because the round-trip delay, recorded by the receiver, often spikes to 3 microseconds. However, with the burst size of 64-128, the signal is clearly visible atop of the noise as the latency gap between contended and uncontended states of the bank exceeds 1 microsecond.

For the burst size of 64, the throughput of the channel reaches 67Kb/s with accuracy of 73.5%. Compared to the best performance in isolated execution, the throughput decreases by 41%. For the burst size of 128, the transmission accuracy is much higher, reaching 95.4%; however, the resulting channel throughput of 65Kb/s is lower than with a burst size of 64 due to a diminished signaling rate.



(a) Burst size 32

(b) Burst size 64

(c) Burst size 128

Figure 6: Signal observed by the receiver in the presence of network load in the private cluster.

Our experiments suggest that the accuracy drop can be mitigated at the cost of a degradation in the channel throughput. We thus conclude that the channel remains robust in the presence of the network load provided that the size of the bursts is adjusted accordingly.

6.3 Stealthiness

We measure the stealthiness of Bankrupt by transmitting 1s to cause maximum memory bandwidth pressure at the intermediary (because transmitting 0s does not generate traffic). We use Perf [3] to monitor the memory traffic on the intermediary. First, we inspect available Infiniband counters [33], e.g., port_xmit_wait, but none of them reveal the network round-trip delays induced by the channel activity. Second, we use aggregate CPU counters, as, to the best of our knowledge, there are no CPUs that feature CPU counters that account for memory requests at the granularity of memory banks.

Burst	1024	512	256	128	64	32	16
Traffic	1.04	0.97	0.82	0.67	0.48	0.31	0.18

Table 2: Memory traffic (in GB/s) generated by the Bankrupt channel, depending on the burst size, on the intermediary. The maximum bank throughput on this platform is 1.22GB/s.

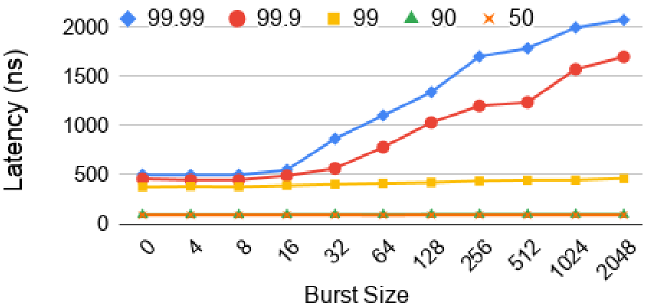


Figure 7: Local memory access latency percentiles recorded on the intermediary during active communication over the Bankrupt channel. Burst size of 0 stands for no active channel. Lines for the 90^{th} and for 50^{th} percentiles appear overlapping.

Table 2 shows the memory traffic that is generated by an active Bankrupt channel on the intermediary. For the burst sizes of 32 and 128, the attack generates memory bandwidth of 0.31GB/s and 0.67GB/s, accordingly. For modern CPUs, which feature 20-32GB/s per channel thanks to the many memory banks, the attack increases their memory bandwidth counters negligibly.

We use a random-access microbenchmark that measures the memory access latency individually using `rdtsc` register with and without the active Bankrupt channel. Figure 7 demonstrates the median and the tail memory latency on our cluster while the channel is not active and while the channel is active with different burst sizes. The 99^{th} percentile increases by less than 10% between the unloaded case and burst size 128. For burst size 32, the 99.9^{th} and the 99.99^{th} percentiles increase by approximately 20% and 70%, respectively. For burst sizes of 512 and larger, the 99.9^{th} and 99.99^{th} percentiles increase from 200% to 400%, but such large bursts are not necessary to transmit a message, as we show in §6.1.

Overall, the Bankrupt attack influences only very high percentiles of the memory access time, and injects negligible memory traffic. Capturing such subtle differences with software is challenging even in the absence of other workloads scheduled on the intermediary's CPU, or other types of the system noise, that renders the attack virtually undetectable.

6.4 Public Cloud Performance

Similarly to the experiments in the private cluster, we identified that the bank bits in CloudLab's servers are located at the positions of bits [6:27] in the address.

As expected for a public cloud, we found that CloudLab's network is more noisy than the private cluster in isolation (§6.1), however less noisy than in the adversarial scenario where we inject network noise with a network loader benchmark (§6.2.2). To stabilize the signal for efficient decoding in CloudLab, we use a smaller 100-bit payload size while keeping the preamble of the same 32-bit size.

Figure 4 shows that the channel achieves ~35% less channel throughput than in the private cluster with almost the same accuracy. This bandwidth reduction can be attributed to two factors: first, the network round-trip in CloudLab is larger due to its scale, and, second, we reduced the payload size in a packet, devoting 13% more of the channel's raw bandwidth for transmitting headers (i.e., preambles). We find that the minimal burst size that allows decoding is 32, which is equal to the minimum burst size in our experiments in the private cluster. With the burst size of 32, Bankrupt provides the channel throughput of 74Kb/s (the accuracy is 82.2%). Increasing the burst size to 128 allows to increase the transmission accuracy to 96.8% while providing 51Kb/s throughput.

7 Detection and Mitigation

Bankrupt does not impact the local memory access time, up to the 99^{th} percentile (§6.3). Available CPU counters account memory requests per memory controller, which is not sufficiently fine-grained to reliably detect memory traffic spikes of <1GB/s directed at just one memory bank. We propose adding per-bank CPU counters to detect the load skew at a finer granularity. However, even though these counters are likely to reveal the attack happening, they would not allow to track down the source of the attack because many applications within the same RDMA network may have access to the remote memory on the intermediary node simultaneously.

We anticipate that architectural changes of the CPU are required to close the Bankrupt channel. As we showed in §6, the Bankrupt channel is robust to different types of noise; as a result, noise injection is unlikely to be an effective preventive measure. Instead, we suggest eliminating the root cause of the vulnerability that is the static memory interleaving scheme that RDMA exposes to an attacker. Hence, CPU architects may consider using a cryptographic block-cipher function, that a memory controller can use for routing memory requests to memory banks, instead of the static one. Using a cryptographic function would significantly complicate the search of addresses that belong to one bank (§4.2.1). Prior work demonstrate that a block-cipher ASIC can perform an encryption operation (e.g., for routing memory requests) in as little as 10ns, and integrating these ASICs inside memory controllers would come at the cost of <0.1% silicon area for a modern CPU [9]. Compared to the memory access latency of a modern CPU, which is normally slightly over 100ns [1], we anticipate a moderate performance impact both for the software that runs locally on the CPU and the RDMA network latency.

8 Responsible Disclosure & Code Availability

We responsibly disclosed the vulnerability to Intel on January 27th 2020, and provided our proof-of-concept code to their security team that confirmed that "an adversary can exfiltrate information leveraging the Bankrupt attack". However, Intel's response highlighted that Intel provide no security guarantees if RDMA-equipped servers are available for use of untrusted parties, i.e., in a public cloud. We disclosed our work to Microsoft Azure's security team that "determined that the attack does not pose a risk to Azure infrastructure due to their architectural decisions". The proof-of-concept source code is published at https://github.com/ease-lab/bankrupt.

9 Related Work

Single-CPU covert channels: Maurice et al. [32] thoroughly studies the peculiarities of cache-based covert channels and demonstrates necessary communication protocol support for robust data transmission. Wu et al. [53] constructs a robust covert channel between virtual machines on different cores of the same node by locking the memory bus with atomic operations. The DRAMA covert channel is based on the timing difference of row buffer hits and misses at DRAM memory banks that can be measured if the sender and the receiver share the same CPU [43]. However, this timing difference (~30ns for DRAM memory) is insufficient to form a robust signal that is visible across a network. To achieve reliable cross-network communication, Bankrupt builds up queuing inside a single memory bank with bursts of row buffer misses, elevating the timing gap to microseconds that is widely visible across an RDMA network. Similar to this work, DRAMA reverse-engineers, using a brute-force search approach, the exact positions and functions of all the bits in a memory address but does so by running software on the target node directly. In contrast, our method allows to retrieve the positions of all the bits that identify memory banks, in a time linear in the number of address bits that is fast in practice, which is necessary to locate addresses in memory banks, across an RDMA network. Contrary to the above memory-based channels, the Bankrupt timing channel relies on inducing queuing effects in the per-bank queues inside a memory controller.

Masti et al. leverage the heat that CPU cores emit to establish covert communication [30]. Xiao et al. exploit memory deduplication to construct a covert channel in a virtualized environment [54, 55]. Other works rely on accesses to private [42] and shared CPU caches [29, 31], including the use of `clflush` instructions [17]. In contrast to all these channels, Bankrupt overcomes the requirement of colocating the sender and the receiver on the same CPU, enabling covert communication across an RDMA network.

Cross-network covert channels: Ovadya et al. exploit the design of network protocols such as ARP, SSH, an ICMP to construct covert channels in routers in TCP/IP networks [41]. Tahir et al. design a covert channel exploiting network links and routers sharing across virtual networks [48]. Both of these channels deliver transmission bandwidth below 2 Kb/s in contrast to 99Kb/s provided by Bankrupt. Recent works have discovered covert channels in modern RDMA deployments. Pythia [49] demonstrates a timing channel in the NIC-internal translation buffer. The channel requires the use of small (e.g., 4KB) pages for the RDMA-exposed regions while the use of large pages is widespread in settings where low latency is a priority [13, 18, 40, 51, 59]. The use of large pages immediately exposes the system to the Bankrupt attack. NetCAT designs a covert channel by leveraging Intel Data-Direct I/O (DDIO) [24] that allows buffering of RDMA packets in the destination's CPU's last-level cache (LLC) [27]. Contrary to NetCAT, Bankrupt cannot be mitigated by disabling DDIO as its timing channel resides in the memory controller. NetCAT delivers similar throughput and accuracy as Bankrupt but lacks sufficient evaluation in the presence of noise. Bankrupt is able to deliver comparable throughput even in an noisy environment thanks to the Bankrupt channel's microsecond-scale latency gap between transmitted `0`-s and `1`-s, which is an order of magnitude larger than the LLC miss delay in NetCAT.

10 Conclusion

Covert channels enable exfiltration of sensitive data by bypassing information containment measures from secure cloud environment to the outside world. From the attacker's perspective, an ideal covert channel should be general enough to unlock high-rate data transfers of an arbitrary size within a datacenter while remaining undetectable from a cloud vendor's monitoring capabilities.

Our work introduces Bankrupt, a high-rate cross RDMA-network covert channel, that meets all of these requirements, by allowing the spy (sender) and the receiver malware, running on different nodes in the network, to communicate via the remote memory that is hosted on yet another – innocuous – node in the same network. We showcase that Bankrupt delivers the channel throughput of 74Kb/s in a large-scale public cloud environment inside the Cloudlab datacenter facility, and up to 114Kb/s in the private cluster. We demonstrate that Bankrupt remains highly robust even in a noisy environment while remaining stealthy to anomaly monitoring capabilities, like Infiniband NIC and CPU counters.

Acknowledgements

The authors thank Prof. Mathias Payer for his valuable feedback on this work as well as the EASE lab members at the University of Edinburgh for the numerous discussions that gave inspiration to this work. The research was supported by the ARM Center of Excellence at the University of Edinburgh.

References

[1] 7-Zip LZMA Benchmark. Available at https://www.7-cpu.com.

[2] CloudLab. Available at https://www.cloudlab.us.

[3] Linux Perf tools. Available at https://man7.org/linux/man-pages/man1/perf.1.html.

[4] Alibaba. Alibaba builds high-speed RDMA network for AI and scientific computing, 2019. Available at https://www.alibabacloud.com/blog/alibaba-builds-high-speed-rdma-network-for-ai-and-scientific-computing_594895.

[5] AMD. AMD EPYC 7002 Series Processors. Available at https://www.amd.com/en/processors/epyc-7002-series.

[6] AMD. BIOS and Kernel Developer's Guide for AMD Family 16h Models 30h-3Fh Processors. Available at https://www.amd.com/system/files/TechDocs/52740_16h_Models_30h-3Fh_BKDG.pdf.

[7] AWS. Placement groups. Available at https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/placement-groups.html.

[8] Azure. Announcing the general availability of proximity placement groups. Available at https://azure.microsoft.com/en-us/blog/announcing-the-general-availability-of-proximity-placement-groups.

[9] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The skinny family of block ciphers and its low-latency variant mantis. In *Annual International Cryptology Conference*. Springer, 2016.

[10] Xiaoqi Chen, Shir Landau Feibish, Yaron Koral, Jennifer Rexford, and Ori Rottenstreich. Catching the microburst culprits with snappy. In *SelfDN colocated with SIGCOMM*, 2018.

[11] Cybersecurity Insiders. Top 5 cloud security related data breaches! Available at https://www.cybersecurity-insiders.com/top-5-cloud-security-related-data-breaches.

[12] Richard Cziva, Christopher Lorier, and Dimitrios P. Pezaros. Ruru: High-speed, flow-level latency measurement and visualization of live internet traffic. In *SIGCOMM*, 2017.

[13] Aleksandar Dragojević, Dushyanth Narayanan, Miguel Castro, and Orion Hodson. FaRM: Fast remote memory. In *NSDI*, 2014.

[14] Dmitry Duplyakin, Robert Ricci, Aleksander Maricq, Gary Wong, Jonathon Duerig, Eric Eide, Leigh Stoller, Mike Hibler, David Johnson, Kirk Webb, et al. The design and operation of CloudLab. In *USENIX ATC*, 2019.

[15] Forbes. EasyJet hacked for four months, data on 9 million customers and 2,000 credit cards stolen. Available at https://www.forbes.com/sites/thomasbrewster/2020/05/19/easyjet-hacked-9-million-customers-and-2000-credit-cards-hit.

[16] Fortune. LinkedIn lost 167 million account credentials in data breach. Available at https://fortune.com/2016/05/18/linkedin-data-breach-email-password.

[17] Daniel Gruss, Clémentine Maurice, Klaus Wagner, and Stefan Mangard. Flush+Flush: A fast and stealthy cache attack. In *DIMVA*, 2016.

[18] Chuanxiong Guo, Haitao Wu, Zhong Deng, Gaurav Soni, Jianxi Ye, Jitu Padhye, and Marina Lipshteyn. RDMA over commodity ethernet at scale. In *SIGCOMM*, 2016.

[19] Health IT Security. The 10 biggest healthcare data breaches of 2019, so far. Available at https://healthitsecurity.com/news/the-10-biggest-healthcare-data-breaches-of-2019-so-far.

[20] Healthcare IT News. Data on 150,000 patients exposed in another misconfigured AWS bucket. Available at https://www.healthcareitnews.com/news/data-150000-patients-exposed-another-misconfigured-aws-bucket.

[21] John L. Henning. SPEC CPU2006 benchmark descriptions. *ACM SIGARCH Computer Architecture News*, 2006.

[22] Huawei. Huawei launches Mellanox-based InfiniBand EDR 100 Gbps switch solution, 2016. Available at https://www.huawei.com/en/press-events/news/2016/6/Mellanox-Based-InfiniBand-EDR-100Gbps-Switch-Solution.

[23] Information Age. Addressing the issue of data leakage from the cloud. Available at https://www.information-age.com/addressing-data-leakage-cloud-123486781.

[24] Intel. Intel Data Direct I/O Technology (Intel DDIO): A Primer. Available at https:

//www.intel.com/content/dam/www/public/us/en/documents/technology-briefs/data-direct-i-o-technology-brief.pdf.

[25] Aamer Jaleel. Memory characterization of workloads using instrumentation-driven simulation. Technical report, 2010. Available at http://www.jaleels.org/ajaleel/publications/SPECanalysis.pdf.

[26] Raj Joshi, Ting Qu, Mun Choon Chan, Ben Leong, and Boon Thau Loo. BurstRadar: Practical real-time microburst monitoring for datacenter networks. In *APSys*, 2018.

[27] Michael Kurth, Ben Gras, Dennis Andriesse, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. NetCAT: Practical cache attacks from the network. In *S&P*, 2020.

[28] Butler W. Lampson. A note on the confinement problem. In *CACM*, 1973.

[29] Fangfei Liu, Yuval Yarom, Qian Ge, Gernot Heiser, and Ruby B. Lee. Last-level cache side-channel attacks are practical. In *S&P*, 2015.

[30] Ramya Jayaram Masti, Devendra Rai, Aanjhan Ranganathan, Christian Müller, Lothar Thiele, and Srdjan Capkun. Thermal covert channels on multi-core platforms. In *USENIX Security*, 2015.

[31] Clémentine Maurice, Christoph Neumann, Olivier Heen, and Aurélien Francillon. C5: Cross-cores cache covert channel. In *DIMVA*, 2015.

[32] Clémentine Maurice, Manuel Weber, Michael Schwarz, Lukas Giner, Daniel Gruss, Carlo Alberto Boano, Stefan Mangard, and Kay Römer. Hello from the other side: SSH over robust cache covert channels in the cloud. In *NDSS*, 2017.

[33] Mellanox. Understanding mlx5 linux counters and status parameters. Available at https://community.mellanox.com/s/article/understanding-mlx5-linux-counters-and-status-parameters.

[34] Mellanox. Mellanox Perftest package, 2017. Available at https://community.mellanox.com/s/article/perftest-package.

[35] Micron. Micron DDR2 SDRAM datasheet. Available at http://www.micron.com/-/media/documents/products/data%20sheet/dram/ddr2/2gb_ddr2.pdf.

[36] Micron. Micron DDR4 SDRAM datasheet. Available at https://www.micron.com/-/media/client/global/documents/products/data-sheet/dram/ddr4/8gb_ddr4_sdram.pdf.

[37] Microsoft. SMB-Direct. Available at https://docs.microsoft.com/en-us/windows-server/storage/file-server/smb-direct.

[38] Microsoft. Availability of Linux RDMA on Microsoft Azure, 2015. Available at https://azure.microsoft.com/en-gb/blog/azure-linux-rdma-hpc-available.

[39] Microsoft. Introducing the new HBv2 Azure virtual machines for high-performance computing, 2019. Available at https://azure.microsoft.com/en-us/blog/introducing-the-new-hbv2-azure-virtual-machines-for-high-performance-computing.

[40] Stanko Novakovic, Yizhou Shan, Aasheesh Kolli, Michael Cui, Yiying Zhang, Haggai Eran, Boris Pismenny, Liran Liss, Michael Wei, Dan Tsafrir, and Marcos Aguilera. StoRM: A fast transactional dataplane for remote data structures. In *SYSTOR*, 2019.

[41] Adar Ovadia, Rom Ogen, Yakov Mallah, Niv Gilboa, and Yossi Oren. Cross-router covert channels. In *WOOT colocated with USENIX Security*, 2019.

[42] Colin Percival. Cache missing for fun and profit. In *BSDCan*, 2005.

[43] Peter Pessl, Daniel Gruss, Clémentine Maurice, Michael Schwarz, and Stefan Mangard. DRAMA: Exploiting DRAM addressing for cross-CPU attacks. In *USENIX Security*, 2016.

[44] Diana Andreea Popescu and Andrew W. Moore. PTPmesh: Data center network latency measurements using PTP. In *MASCOTS*, 2017.

[45] Diana Andreea Popescu and Andrew W. Moore. A first look at data center network condition through the eyes of PTPmesh. In *TMA*, 2018.

[46] Joel Reardon, Álvaro Feal, Primal Wijesekera, Amit Elazari Bar On, Narseo Vallina-Rodriguez, and Serge Egelman. 50 ways to leak your data: An exploration of apps' circumvention of the Android permissions system. In *USENIX Security*, 2019.

[47] RedHat. NFS over RDMA. Available at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/storage_administration_guide/nfs-rdma.

[48] Rashid Tahir, Mohammad Taha Khan, Xun Gong, Adnan Ahmed, Amiremad Ghassami, Hasanat Kazmi, Matthew Caesar, Fareed Zaffar, and Negar Kiyavash. Sneak-peek: High speed covert channels in data center networks. In *INFOCOM*, 2016.

[49] Shin-Yeh Tsai, Mathias Payer, and Yiying Zhang. Pythia: Remote oracles for the masses. In *USENIX Security*, 2019.

[50] UpGuard. What are cloud leaks? Available at https://www.upguard.com/blog/what-are-cloud-leaks.

[51] Zhi Wang, Xiaoliang Wang, Zhuzhong Qian, Baoliu Ye, and Sanglu Lu. RDMAvisor: Toward deploying scalable and sample RDMA as a service in datacenters, 2018. Available at http://arxiv.org/abs/1802.01870.

[52] Wired. Hack Brief: 4-year-old Dropbox hack exposed 68 million people's data. Available at https://www.wired.com/2016/08/hack-brief-four-year-old-dropbox-hack-exposed-68-million-peoples-data.

[53] Zhenyu Wu, Zhang Xu, and Haining Wang. Whispers in the hyper-space: High-speed covert channel attacks in the cloud. In *USENIX Security*, 2012.

[54] Jidong Xiao, Zhang Xu, Hai Huang, and Haining Wang. A covert channel construction in a virtualized environment. In *CCS*, 2012.

[55] Jidong Xiao, Zhang Xu, Hai Huang, and Haining Wang. Security implications of memory deduplication in a virtualized environment. In *DSN*, 2013.

[56] Yuval Yarom and Katrina Falkner. FLUSH+RELOAD: A high resolution, low noise, L3 cache side-channel attack. In *USENIX Security*, 2014.

[57] ZDNet. Hacker leaks passwords for more than 500,000 servers, routers, and IoT devices. Available at https://www.zdnet.com/article/hacker-leaks-passwords-for-more-than-500000-servers-routers-and-iot-devices.

[58] Yunqi Zhang, David Meisner, Jason Mars, and Lingjia Tang. Treadmill: Attributing the source of tail latency through precise load testing and statistical inference. In *ISCA*, 2016.

[59] Peipei Zhou, Zhenyuan Ruan, Zhenman Fang, Megan Shand, David Roazen, and Jason Cong. Doppio: I/O-aware performance analysis, modeling and optimization for in-memory computing framework. In *ISPASS*, 2018.