Calculated Compromise: Characterizing Cloud-Based Cyber Conflicts

Katrina Nicole Natura

Western Governors University

WESTERN GOVERNORS UNIVERSITY

Table of Contents

Summary 3
Review of Other Work 4
Work 1: T-Pot Official Documentation 4
Work 2: Elastic/Kibana Documentation 5
Work 3: OpenSCAP Documentation 6
Changes to the Project Environment 7
Methodology 8
Phase 1: Plan 8
Phase 2: Do (Infrastructure Provisioning and Deployment) 9
Phase 3: Check (Threat Intelligence Analysis) 10
Phase 4: Act (Hardening and Validation) 10
Project Goals and Objectives 11
Project Timeline 13
Project Timeline Table 14
Unanticipated Scope Creep 14
Conclusion 16
References 19
Appendix A 22
Creation and Deployment of Azure Virtual Machine with T-Pot 22
Appendix B 31
T-Pot Attack Analysis and Data Visualization 31
Appendix C 38
Secured Virtual Machine Hardening and Compliance Validation 38

Summary

The project addressed a critical visibility gap related to “opportunistic” threat activity within a Microsoft Azure cloud environment. Specifically, the project examined a simulated small-to-medium enterprise (SME) service hosted entirely in Azure, serving fewer than 7,000 monthly visitors and relying on standard security practices. Prior to the implementation of the proposed solution, the simulated service provider relied exclusively on Microsoft Azure Network Security Groups (NSGs) and basic firewall controls -- lacking insight into malicious targeted activity. This absence of contextual intelligence left the simulated production asset (the “Secured VM”) vulnerable to emerging threats such as credential stuffing and automated exploit attempts, as the organization had no mechanism to proactively identify or block high-risk vectors before exploitation attempts occurred.

The project was executed using the PDCA (Plan–Do–Check–Act) methodology, with implementation divided into clear, consecutive phases. Microsoft Azure was selected as the hosting environment, and the T-Pot Community Edition platform was chosen for its multi-sensor honeypot architecture. A dedicated Azure Resource Group was provisioned, within which a Debian 12 virtual machine was deployed to host the T-Pot “Hive” sensor array alongside a separate simulated Service VM. The honeypot environment was left operational for a seven-day data collection period, during which attack telemetry was passively gathered from the internet. Collected data was analyzed using the Kibana dashboard, and the resulting threat intelligence was used to create and apply targeted security rules to harden the secured web service virtual machine.

The project successfully generated actionable threat intelligence that was previously unavailable to the simulated service provider. During the collection window, the honeypot recorded thousands of unauthorized interaction attempts, revealing consistent automated attack behavior across multiple protocols. Key outcomes included the identification of recurring botnet activity, commonly attempted

credentials, and exploit payloads targeting known web-related vulnerabilities. An important finding of the project was that remediation actions derived from honeypot intelligence did not produce a measurable change in the system's OpenSCAP CIS benchmark audit score. This outcome demonstrates that while CIS benchmarks provide a valuable baseline for system hardening, they do not account for environment-specific threat intelligence or adaptive defensive controls. As a result, the project highlights the necessity of supplementing baseline compliance frameworks with intelligence-driven security measures tailored to the unique threat landscape facing a given environment.

Review of Other Work

To support the specific technical implementation and data analysis phases of this project, three new works were consulted. These sources were determined to be reputable specifically for reference during implementation, troubleshooting and referencing best known practices during the execution of the project.

Work 1: T-Pot Official Documentation

The official GitHub repository containing the T-Pot distribution was extensively referenced throughout the project. The repository is maintained by Deutsche Telekom Security, a reputable telecommunications IT services and consulting organization and an Akamai Partner, which reinforces the credibility of the platform and its supporting documentation (Deutsche Telekom Security, n.d.; Akamai, 2023). The availability of a free, enterprise-developed deception framework further supported its selection as a viable solution for this project.

Reviewing this guide was instrumental in understanding T-Pot's hive architecture and how its components operate together within a single deployment. T-Pot leverages Docker to simultaneously deploy multiple honeypot sensors, enabling broad protocol coverage while maintaining isolation between services. These sensors operate alongside Suricata, a network security monitoring engine that

performs deep packet inspection and traffic fingerprinting by analyzing packet payloads to identify known malicious patterns (Open Information Security Foundation, n.d.). To enrich this network telemetry, the platform integrates SpiderFoot, an open-source intelligence (OSINT) automation tool that correlates locally observed attacker data with external reputation and threat intelligence sources (Micallef, n.d.). All collected data is aggregated through the Elastic Stack and made accessible through Kibana dashboards, with Elasticvue providing a lightweight interface for direct Elasticsearch cluster management to ensure the captured intelligence is structured and accessible for analysis (Carsten, n.d.).

The T-Pot documentation was used to determine the technical requirements necessary for implementation, including supported Linux distributions, required open ports, hardware and processing recommendations, and installation procedures. These instructions were directly referenced during deployment of the T-Pot "Hive," troubleshooting connectivity issues, validating web interface access, and importing and exporting Kibana objects. Additionally, the documentation provided guidance on troubleshooting steps such as verifying container health, identifying port conflicts, and confirming that system resources were not exhausted, all of which were applied during project execution (Deutsche Telekom Security, n.d.).

The guide also influenced critical infrastructure decisions related to resource allocation. Specifically, the documentation states that "the more honeypots, sensors & data, the more RAM and storage is needed," and further notes that "some users report working installations on other clouds… hardware requirements may be different" (Deutsche Telekom Security, n.d.). These recommendations directly informed the decision to allocate 32 GB of RAM to the Azure-hosted honeypot virtual machine.

Work 2: Elastic/Kibana Documentation

The official Elastic documentation on Kibana dashboards and controls was reviewed to establish a technical understanding of how honeypot telemetry collected by T-Pot could be effectively analyzed

and operationalized. As documentation published directly by Elastic N.V., the developers of the Elastic Stack, this source represents the authoritative reference for implementing visualization, filtering, and query functionality within Kibana (Elastic, n.d.-a; Elastic, n.d.-b). This work directly supported the project by defining how raw attack data could be transformed into structured, actionable intelligence.

Specifically, this documentation provided guidance on the use of Kibana Query Language (KQL), filter pills, and interactive dashboard controls to segment high-volume datasets. These techniques were applied during the analysis phase to isolate malicious traffic from background internet noise by filtering on source IP reputation, destination ports, protocols, and user-agent strings. The documented use of Global Time Range filtering was also directly applied to constrain analysis to the defined seven-day collection window, ensuring that the visualizations accurately reflected the active attack period rather than cumulative historical data.

Additionally, this work informed considerations for deploying the honeypot in a real operational environment. The documentation describes advanced dashboard features, including variable controls bound to Elasticsearch queries, which enable analysts to dynamically adjust parameters such as aggregation intervals and filtered values without modifying the underlying queries (Elastic, n.d.-b). This capability aligns directly with the project's objective of supporting iterative threat investigation, as it would allow security teams to rapidly pivot between attack patterns, credential attempts, and targeted assets using the same collected intelligence.

Work 3: OpenSCAP Documentation

The official OpenSCAP documentation was referenced to establish the procedural methodology for installing and executing the OSCAP scanner on the Secured VM infrastructure. As the official technical guide from the OpenSCAP project—an ecosystem aligned with NIST standards and widely adopted for automated compliance checking—this source is inherently credible and authoritative

(OpenSCAP, n.d.). It directly supported the project by providing the required baseline compliance workflow for validating the security posture of the simulated service VM.

Reviewing this documentation clarified the exact command-line syntax needed to install OpenSCAP libraries and the CIS benchmark content on Ubuntu systems. It expanded on the structure of the evaluation command (**oscap xccdf eval**), including how to specify input profiles and output formats to generate the HTML artifacts used in Appendix C (OpenSCAP, n.d.). This guidance was directly applied when creating the audit script for Ubuntu 22.04 using the CIS Level 1 benchmark profile, ensuring the baseline scan aligned with the correct version-specific standard.

Additionally, the Ubuntu 22.04 CIS guide provided detailed remediation logic for high-severity findings identified during the initial scan. This documentation was referenced during the hardening phase to adjust SSH configuration settings—specifically restricting root login and empty passwords—which directly supported the manual remediation efforts and improved the OpenSCAP compliance score from 66.38% to 68.23% (OpenSCAP, n.d.-b).

Changes to the Project Environment

The operational environment has undergone a fundamental shift from a reactive to a threat-informed security posture—shifting it's exclusive reliance on generic cloud-native controls and standard compliance baselines. The successful deployment of the T-Pot honeypot has changed this ecosystem by introducing an active "sensor node" within the network, allowing the organization to secure its production assets based on verifiable, real-time attack data rather than theoretical best practices alone.

Strategically, this project mandates a move away from "set-and-forget" firewall configurations toward an Active Defense strategy. The successful identification of targeted threats—such as specific VNC exploits and python-requests callbacks—demonstrates the necessity of configuring the Elastic Stack

(ELK) to alert specifically on high-fidelity threats that target the organization's unique data structure. This ensures that defensive resources are prioritized against actual automated scan activity and targeted campaigns rather than generic background noise.

Culturally, the organization must adopt a Continuous Security Posture Optimization cycle—requiring a "Cyclic" operational model where the honeypot is continuously maintained to capture evolving threats. The insights gained from this deceptive layer must drive an iterative feedback loop: constantly updating the honeypot to mimic new assets, analyzing the resulting breach data, and immediately applying those findings to harden the production environment. This shift aligns the organizational culture with a DevSecOps mindset, where security is an ever-evolving process of measurement, analysis, and adaptation.

Methodology

This project was executed using the Plan–Do–Check–Act (PDCA) methodology because it supports iterative improvement, evidence-based decision-making, and validation of applied controls. Each phase of the methodology is described below as it was applied during the implementation of the project.

Phase 1: Plan

The Plan phase focused on designing a cloud-based architecture capable of capturing high-fidelity threat intelligence while supporting later defensive validation. Microsoft Azure was selected as the hosting platform to ensure exposure to realistic cloud-based attack traffic. A dual–virtual machine architecture was planned, consisting of a dedicated honeypot virtual machine and a separate “Secured VM” simulating a production web service.

During this phase, research was conducted to determine the optimal operating system and hardware specifications required to support the T-Pot "Hive" deployment. Debian 12 was selected as the operating system for the honeypot virtual machine due to its compatibility with the T-Pot platform and containerized services. Hardware planning initially targeted the minimum recommended requirements; however, research and early testing revealed that the Elastic Stack component of T-Pot was unstable with minimal memory allocation. As a result, the plan was adjusted to provision the honeypot with 32 GB of RAM and a 128 GiB operating system disk to ensure reliable data ingestion and analysis. Additionally, OpenSCAP was selected during this phase as the auditing tool to evaluate the security posture of the Secured VM using CIS benchmarks.

Phase 2: Do (Infrastructure Provisioning and Deployment)

The Do phase involved provisioning and configuring the Azure infrastructure and deploying the honeypot environment. The creation of the Azure virtual machines and initial configurations are documented in ***Appendix A, Figures A-1 through A-8***. While the honeypot was initially deployed with 8 GB of RAM, this configuration resulted in system instability and boot-loop behavior. The virtual machine was subsequently resized to 32 GB of RAM.

Secure administrative access was established using SSH, enabling the administrator to remotely connect to the honeypot virtual machine ***(Appendix A, Figure A-9).*** The T-Pot Community Edition repository was cloned from GitHub, and the installation was executed from the /tpotce directory, during which administrative credentials for the web interface were configured ***(Appendix A, Figures A-10 through A-12)***. Successful deployment of the T-Pot 24.04.1 "Hive" stack was confirmed through access to the T-Pot web dashboard via port 64297 ***(Appendix A, Figures A-13 through A-15).***

Network exposure was intentionally configured through Azure Network Security Group (NSG) rules to maximize attack surface visibility. As shown in ***Appendix A, Figure A-16***, administrative access

ports (64295 and 64297) were restricted exclusively to the administrator's IP address, while a lower-priority rule allowed all inbound traffic across all protocols and ports to expose the honeypot sensors to opportunistic attack traffic.

Phase 3: Check (Threat Intelligence Analysis)

During the Check phase, the honeypot was left operational for a seven-day collection period. Attack telemetry was aggregated and analyzed using the integrated Elastic Stack and visualized through Kibana dashboards. ***Appendix B, Figure B-1*** illustrates real-time global attack activity, with interaction intervals frequently occurring within seconds, confirming continuous data collection.

Analysis focused on extracting actionable intelligence from approximately 478,000 recorded events ***(Appendix B, Figure B-2)***. Additionally, applying a filter to analyze activity from IP addresses with a known malicious reputation revealed that the most frequently targeted destination ports were 22, 5060, 8728, 80, and 25 ***(Appendix B, Figure B-3)***. Signature-based detection provided by the Suricata sensor identified the top alert categories, including significant VNC and ICMP activity, as well as more than 8,000 inbound requests associated with the python-requests user agent ***(Appendix B, Figure B-4).*** This behavior indicated automated scripts attempting post-exploitation activity, such as downloading second-stage payloads or establishing command-and-control callbacks.

Additional analysis identified the most aggressive source IP addresses ***(Appendix B, Figure B-5)*** and the most frequently targeted usernames, including root and admin, confirming widespread credential brute-force behavior ***(Appendix B, Figure B-6).***

Phase 4: Act (Hardening and Validation)

The Act phase applied the collected threat intelligence to harden a separate Secured VM deployed in Azure. This system was provisioned with Ubuntu Server 22.04, two virtual CPUs, and 8 GB of

RAM ***(Appendix C, Figure C-1).*** A basic web service was configured to verify external accessibility ***(Appendix C, Figure C-2).***

An initial OpenSCAP CIS benchmark audit established a baseline compliance score of 66.38%, identifying five high-severity failures ***(Appendix C, Figure C-3).*** The first remediation stage addressed standard benchmark failures, including disabling empty-password SSH authentication and preventing root login. These changes were implemented using the Azure Serial Console ***(Appendix C, Figures C-4 and C-5).*** A follow-up audit confirmed remediation of these controls, increasing the compliance score to 68.23% ***(Appendix C, Figure C-6).***

The second remediation stage applied intelligence-driven controls based on honeypot findings. Custom Azure NSG rules were created to block the top malicious source IP addresses identified during the analysis phase ***(Appendix C, Figure C-7).*** Additionally, outbound traffic on ports 80 and 443 was restricted to prevent python-requests–based callback activity, mitigating the risk of post-exploitation malware downloads or command-and-control communication ***(Appendix C, Figure C-8).***

A final OpenSCAP audit was conducted and published to the Secured VM as an HTML report ***(Appendix C, Figure C-9).*** While the compliance score remained unchanged at 68.23%, this result demonstrated that threat-intelligence-driven controls extend beyond what generalized compliance benchmarks are designed to measure. Upon completion of validation, all project resources were deprovisioned and deleted.

Project Goals and Objectives

The project successfully accomplished its primary goal of designing and deploying a cloud-based deception environment capable of collecting, analyzing, and operationalizing real-world threat intelligence. Goal 1: Creation of a Cloud-Based Deception Environment was achieved because the

project resulted in a fully operational honeypot platform deployed within a scalable Microsoft Azure infrastructure. As shown in ***Appendix A***, Objective 1: Scalable Cloud Infrastructure Provisioning was met through the successful creation of a dedicated Azure Resource Group and virtual networking configuration that supported independent honeypot and service workloads. Additionally, Objective 2: T-Pot Framework Deployment was accomplished through the installation and successful operation of the T-Pot "Hive" framework on the CapstoneHoney virtual machine, as evidenced by active services and sensor availability. The ability of the system to accept inbound connections and log interaction data confirms that the deception environment was fully functional and capable of fulfilling its intended role.

Furthermore, Goal 2: Collection and Analysis of Threat Data was also accomplished, as the deployed honeypot actively captured and processed real-world adversary interaction data over the designated collection period. Evidence in ***Appendix B*** demonstrates that Objective 3: Analyze Threat Data was met through the use of Kibana dashboards and filters to examine attack sources, targeted services, and observed behavior patterns. The presence of repeated automated scanning activity, protocol-specific probing, and credential-based attack attempts confirms that the project did not merely collect raw data, but successfully transformed telemetry into interpretable and actionable threat intelligence. This analytical capability directly satisfies the goal of understanding adversary behavior within the specific cloud environment.

Finally, Goal 3: Application of Threat Intelligence Security Insights onto the Service VM was achieved because the analyzed threat data was directly used to inform defensive security controls within the simulated service environment. As documented in ***Appendix C***, intelligence gathered from the honeypot was translated into targeted configuration changes and security rules applied to the Service VM. Objective 2: Verification of the Efficacy of the Security Mechanisms Applied was met through post-remediation OpenSCAP assessments, which confirmed that the applied controls were successfully

implemented and auditable. Although the CIS benchmark score did not significantly change, the results validate that intelligence-driven controls were applied beyond baseline configurations, demonstrating that environment-specific threat mitigation can exist independently of generalized compliance scoring.

Project Timeline

The initial project timeline required adjustment during execution due to unanticipated technical constraints and a deliberate expansion of the threat data collection window. While the Threat Analysis and Secure VM Configuration phases were completed within their originally projected timeframes, the Infrastructure Provisioning and Data Collection phases exceeded initial estimates. These deviations were the result of necessary remediation efforts and methodological decisions made to ensure project stability and data validity rather than project delays caused by poor planning or execution.

The original six-hour estimate for infrastructure provisioning proved insufficient due to critical stability issues encountered during the initial T-Pot deployment. Although the Azure resource group and base virtual machine were provisioned as scheduled, a persistent system boot loop prevented the honeypot from entering a stable operational state. As a result, the infrastructure phase was extended to approximately 48 hours to allow for troubleshooting and remediation. Resolution required the creation of custom reboot scripts, manual restoration of missing library dependencies, vertical scaling of the CapstoneHoney virtual machine to increase available memory, and additional configuration to reestablish reliable data ingestion between the Elastic Stack and Kibana dashboards. These corrective actions were necessary to ensure long-term system reliability before proceeding with data collection.

The originally planned 48-hour data collection window was extended to a seven-day operational period after preliminary telemetry review indicated that the shorter window did not produce a sufficiently representative threat baseline. Expanding the collection timeframe allowed for the

observation of a broader range of global attack sources, recurring behaviors, and automated scanning patterns. This extension reduced the risk of short-term anomalies influencing the analysis and improved the statistical reliability of the collected data. As a result, the extended collection period directly contributed to the quality and applicability of the threat intelligence used in later project phases.

Project Timeline Table

Milestone	Duration	Start Date	End Date	Variance
Infrastructure Provisioning	48 hours	1/13/2026	1/15/2026	+42 hours
Data Collection Period	7 days	1/15/2026	1/22/2026	+5 days
Threat Analysis & Blocklist Generation	3 hours	1/22/2026	1/22/2026	On Target
Secure VM Configuration	12 hours	1/23/2026	1/23/2026	On Target

Unanticipated Scope Creep

Several unanticipated technical issues arose during the deployment and stabilization of the project infrastructure, resulting in scope creep beyond the originally planned configuration tasks. The most significant issue occurred during the deployment of the CapstoneHoney virtual machine, where multiple T-Pot services entered a persistent restart loop. Resolving this issue required additional configurations, repeated system reboots, and partial reinstallation of services that were not accounted for in the original project scope. Root cause analysis revealed a blockage within the Logstash ingestion pipeline, which prevented proper data flow into the Elastic Stack. This issue was resolved by injecting specific CORS configurations into the **elasticsearch.yml** file to restore administrative access, flushing the Logstash queue to remove malformed ingestion data, restarting the container stack, and vertically scaling the VM by increasing available RAM to ensure system stability.

A second major scope expansion occurred during the threat analysis phase when Kibana dashboards were accessible but displayed no populated data, despite the honeypot actively receiving attack traffic. Initial troubleshooting focused on validating that T-Pot was functioning correctly and ingesting telemetry. This was confirmed by verifying container uptime using **docker ps** and querying Elasticsearch directly with **curl -s -XGET http://127.0.0.1:9200/_cat/indices?v**, which showed millions of indexed documents stored within Logstash indices. Although data ingestion was occurring successfully, the visualization layer remained nonfunctional, indicating an aggregation or configuration issue.

Further investigation required the use of Azure's native Command Run feature to execute diagnostic commands directly on the CapstoneHoney VM. Commands such **as find / -name "*dashboard*.json"** and **docker logs tpotinit** were used to locate missing visualization assets and review initialization logs. Analysis of this output revealed that the standard **tpot_dashboards.json** file was no longer present due to changes in the file structure introduced in T-Pot version 24.04, where the dashboard configuration had been renamed and relocated to a distribution directory.

To resolve this issue, additional remediation steps were introduced beyond the original project scope. A current T-Pot Kibana configuration file (**kibana_export.ndjson**) was retrieved from the official GitHub repository in raw format, transferred to the administrator's local machine, and manually imported through Kibana Stack Management > Saved Objects. This action successfully restored the visualization layer and enabled full analysis of the collected threat intelligence. While these issues were ultimately resolved, they required extended troubleshooting time and deeper platform-level investigation, representing unanticipated scope creep that expanded both the technical complexity and duration of the project.

Conclusion

The results of this project successfully demonstrate that honeypot-derived threat intelligence, while often overlooked, is an instrumental component of a robust cloud defense strategy. As observed during industry-wide events like the Bitdefender Log4Shell analysis, honeypots are uniquely capable of revealing adversary scanning for zero-day vulnerabilities that standard preventative tools may miss (Bitdefender, 2021). Unlike traditional preventative controls that rely on static signatures or predefined allow/deny rules, the deployed honeypot captured adversary intent by directly observing real-world attack behavior within the Azure environment. This capability enabled the project to surface attack techniques and patterns that would otherwise remain invisible to the simulated service provider.

Using the evaluation framework defined in Task 2 and validated through the Project Goals and Objectives section of this report, the project was determined to be successful. All primary goals were achieved: a cloud-based deception environment was provisioned, meaningful threat intelligence was collected and analyzed, and the resulting insights were implemented through targeted defensive controls applied to a secured service virtual machine. The successful execution of each objective demonstrated that honeypot intelligence can be translated into actionable security improvements within a cloud-hosted infrastructure.

While the mitigation techniques implemented—such as source IP blocking and outbound traffic restrictions—do not represent a comprehensive or permanent solution due to attacker adaptability, their application demonstrated measurable defensive value. The project confirmed that intelligence-driven security controls enhance resilience beyond what generalized compliance benchmarks alone can provide. Even though these controls did not materially alter baseline OpenSCAP compliance scores, the findings highlight that environment-specific threat intelligence complements, rather than replaces,

traditional security standards. Collectively, the project's results reinforce the practical value of honeypots as an adaptive detection and response mechanism within evolving cloud threat landscapes

References

Akamai. (2021, January 26). Deutsche Telekom Security expands cybersecurity offerings with Akamai. *Akamai Newsroom*. https://www.akamai.com/newsroom/press-release/deutsche-telekom-security-expands-cybersecurity-offerings-with-akamai

Bitdefender. (2021, December 10). Bitdefender honeypots signal active Log4Shell 0-day attacks underway; patch immediately. *Bitdefender Labs*. https://www.bitdefender.com/en-us/blog/labs/bitdefender-honeypots-signal-active-log4shell-0-day-attacks-underway-patch-immediately

Carsten. (n.d.). *Elasticvue: Elasticsearch GUI for the browser*. GitHub. https://github.com/cars10/elasticvue/

Couture, C. (n.d.). Cats. *ASCII Art Archive*. https://www.asciiart.eu/animals/cats

Deutsche Telekom Security. (n.d.). *T-Pot CE discussions: Issue #1810*. GitHub. https://github.com/telekom-security/tpotce/discussions/1810

Deutsche Telekom Security. (n.d.). *Home*. LinkedIn. https://www.linkedin.com/company/telekom-security/

Elastic. (n.d.). *Add controls*. Elastic Documentation. https://www.elastic.co/docs/explore-analyze/dashboards/add-controls

Elastic. (n.d.). *Explore and analyze*. Elastic Documentation. https://www.elastic.co/docs/explore-analyze/

Micallef, S. (n.d.). *SpiderFoot: Automate OSINT*. GitHub. https://github.com/smicallef/spiderfoot

Open Information Security Foundation. (n.d.). *What is Suricata?*. Suricata Documentation. https://docs.suricata.io/en/latest/what-is-suricata.html

OpenSCAP. (n.d.). *Getting started*. OpenSCAP. https://www.open-scap.org/getting-started/

OpenSCAP. (n.d.). *Guide to the Secure Configuration of Ubuntu 22.04 LTS (CIS Level 1 Workstation)*. OpenSCAP Security Guide. https://static.open-scap.org/ssg-guides/ssg-ubuntu2204-guide-cis_level1_workstation.html

OpenSCAP. (n.d.). *Guide to the Secure Configuration of Ubuntu 22.04 LTS (Standard)*. OpenSCAP Security Guide. https://static.open-scap.org/ssg-guides/ssg-ubuntu2204-guide-standard.html

Qexa. (n.d.). *Azure T-Pot honeypot troubleshooting guide*. GitHub. https://github.com/qexa/azure-tpot-honeypot/blob/main/docs/troubleshooting.md

Ubuntu. (n.d.). *Index of /ubuntu/pool/universe/o/openscap/*. Ubuntu Security Repository. http://security.ubuntu.com/ubuntu/pool/universe/o/openscap/

Unixcop. (n.d.). *How to install OpenSCAP on Ubuntu 20.04/22.04 LTS*. Unixcop. https://unixcop.com/how-to-install-openscap-on-ubuntu-20-04-22-04-lts/

Appendix A

Creation and Deployment of Azure Virtual Machine with T-Pot

Figure A-1. Select Create Resource in Microsoft Azure

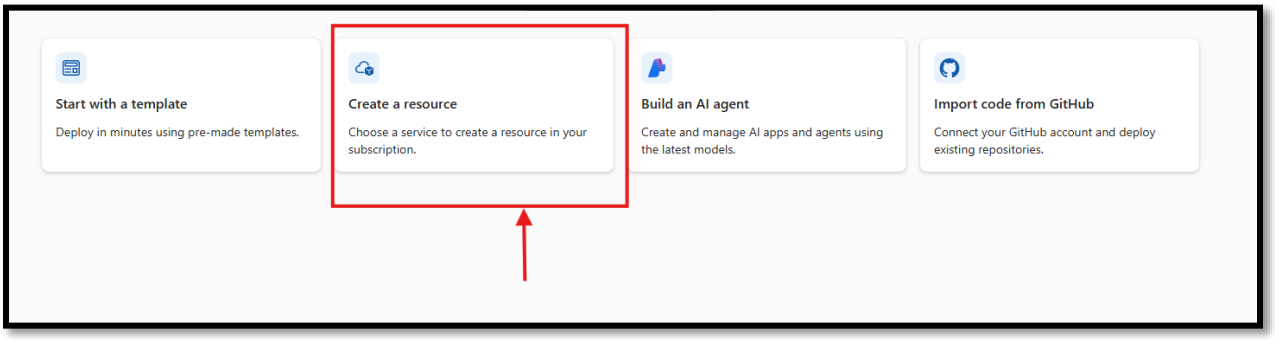


Figure A-2. Select Create Virtual Machine

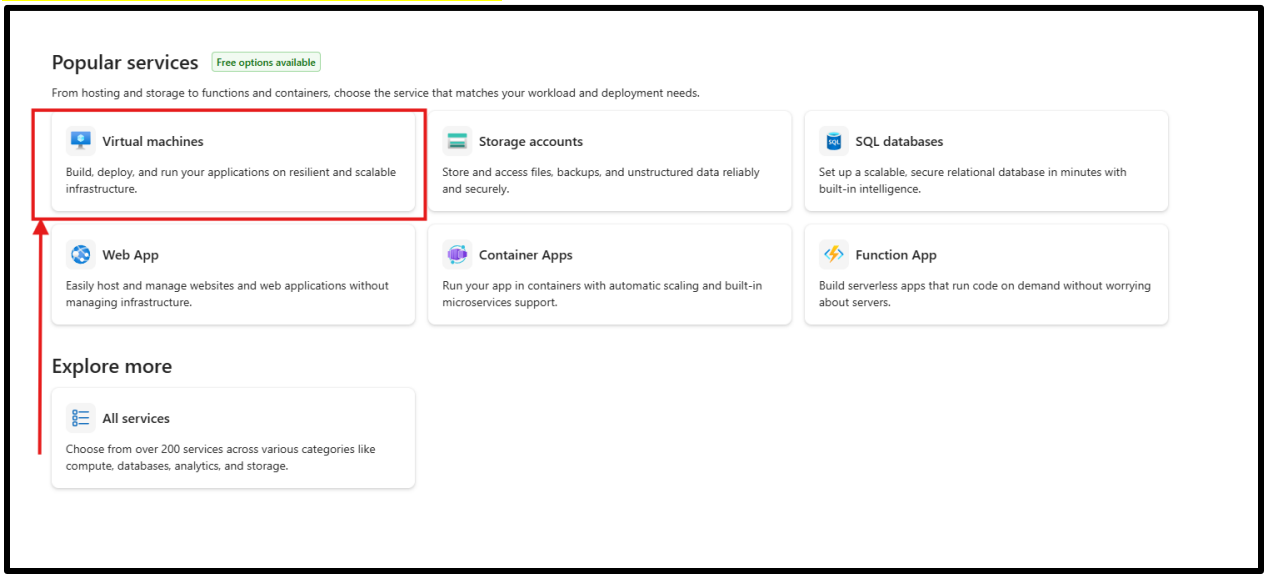


Figure A-3. Configure subscription, resource group, VM name, region, and Debian 12 x64 image

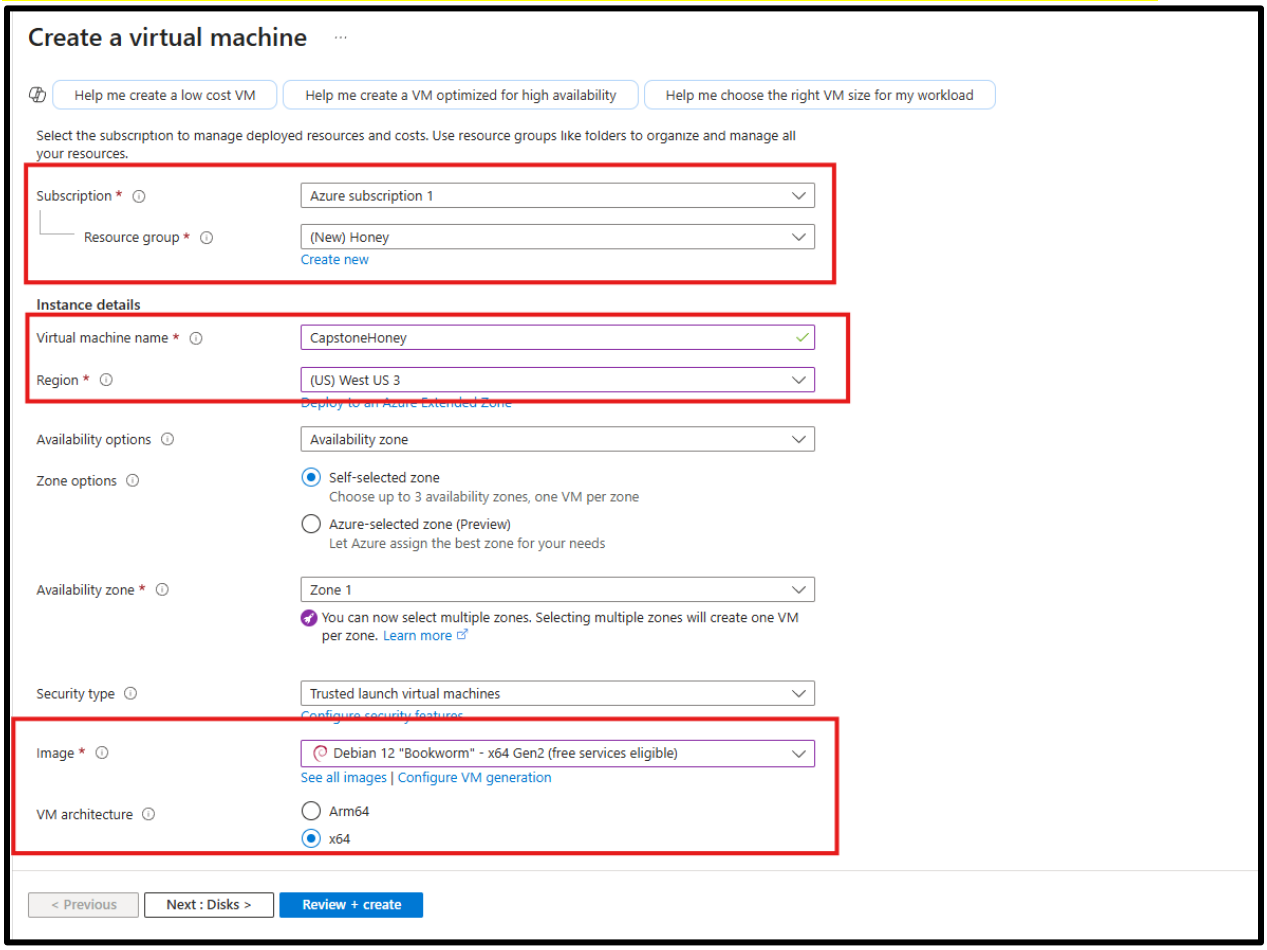


Figure A-4. Set OS disk size to 128 GiB

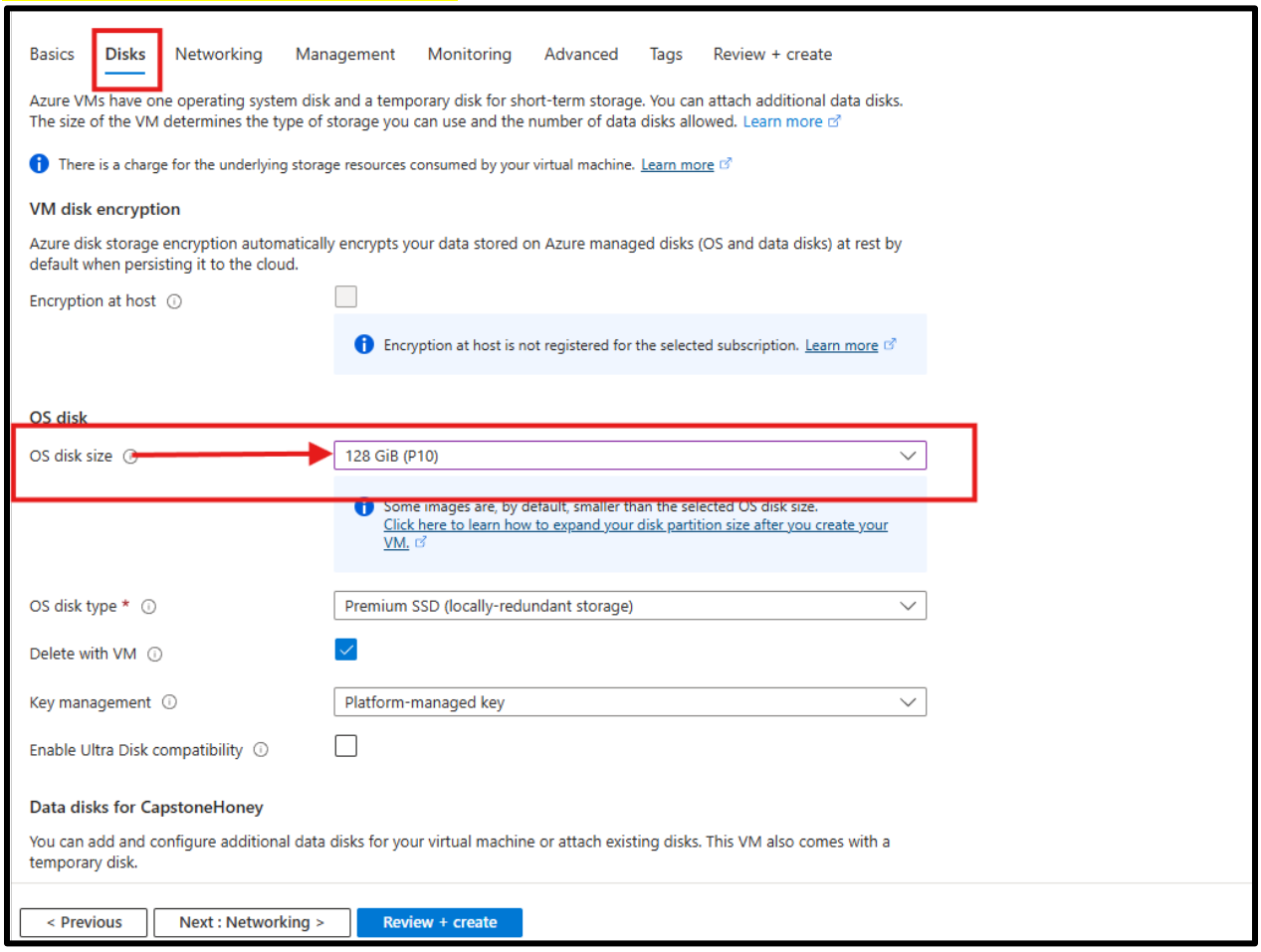


Figure A-5. Ensure Auto-shutdown is unchecked

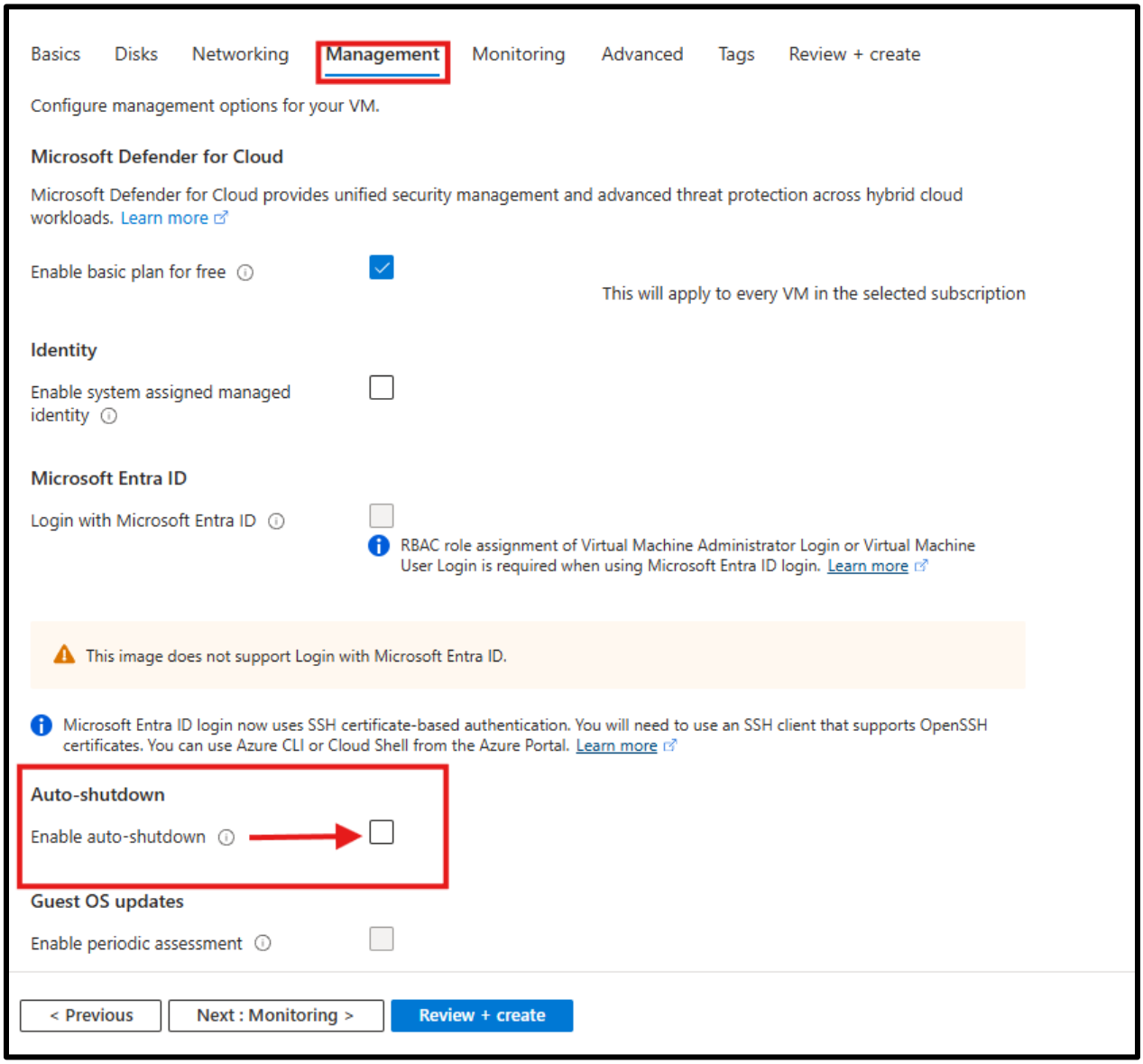


Figure A-6. Review and create virtual machine

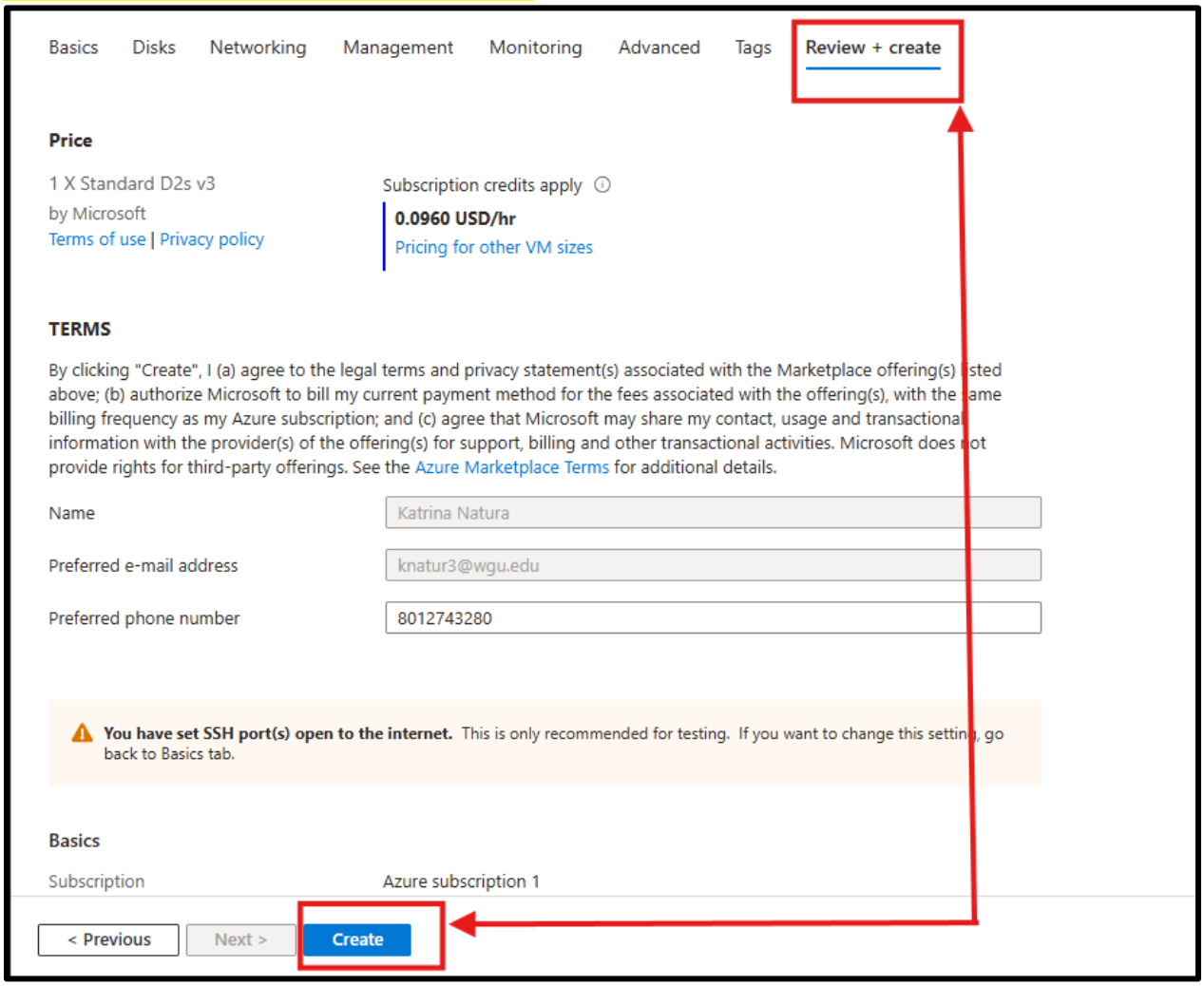


Figure A-7. Download and create private SSH keys

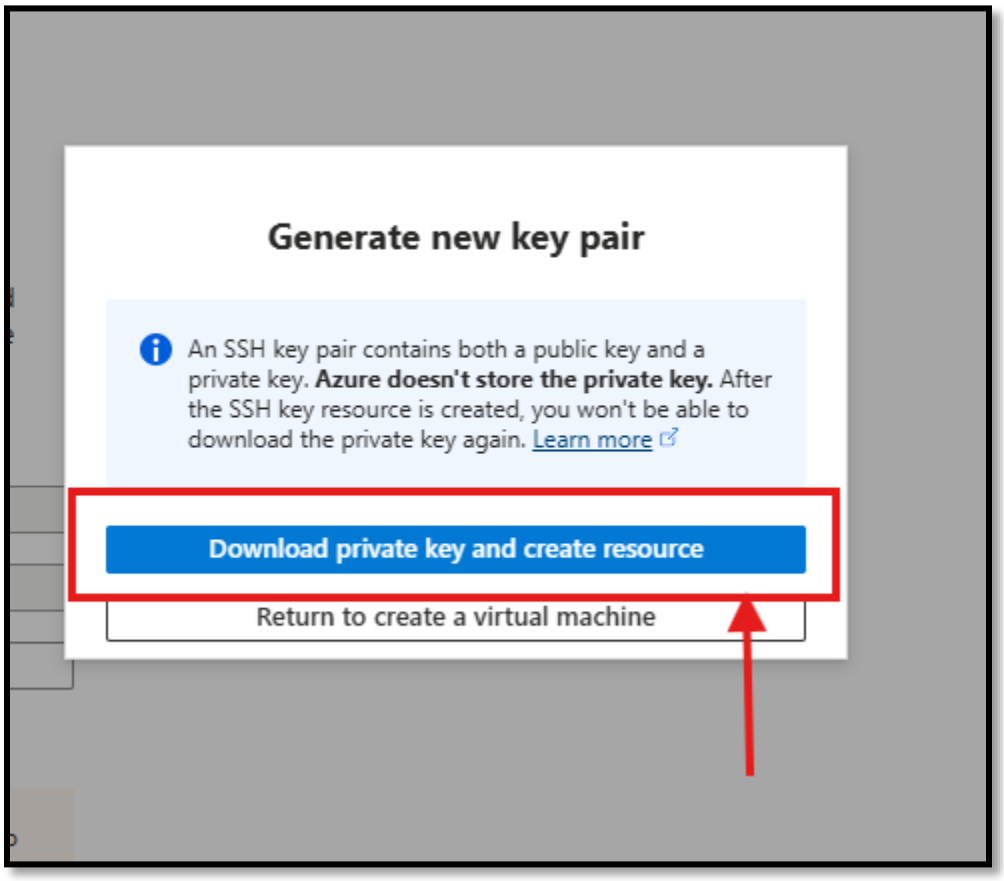


Figure A-8. Virtual machine successfully deployed and active

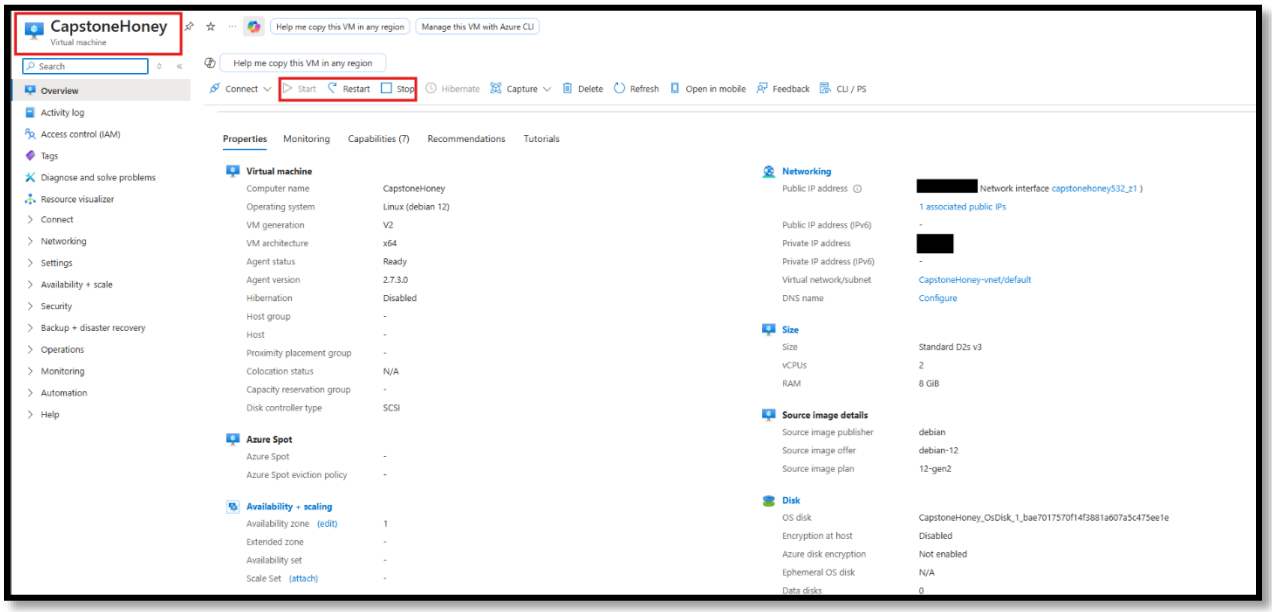


Figure A-9. Successful SSH connection to VM from local machine

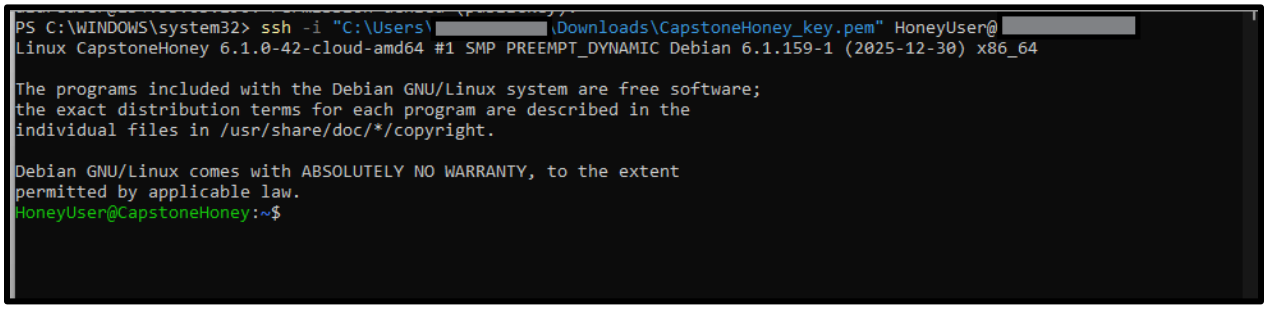


Figure A-10. System maintenance command prior to installing T-Pot

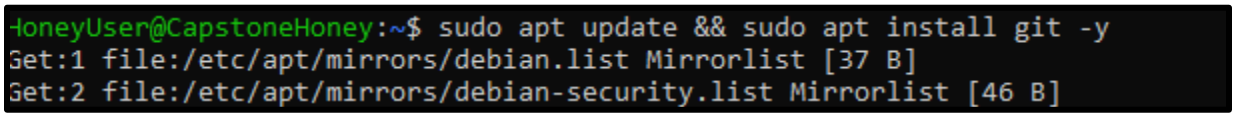


Figure A-11. Downloading T-Pot from the GitHub repository

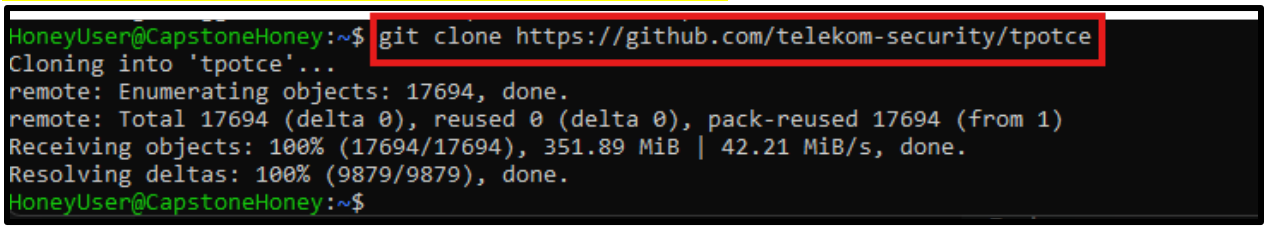


Figure A-12. Running the T-Pot installer from the tpotce directory

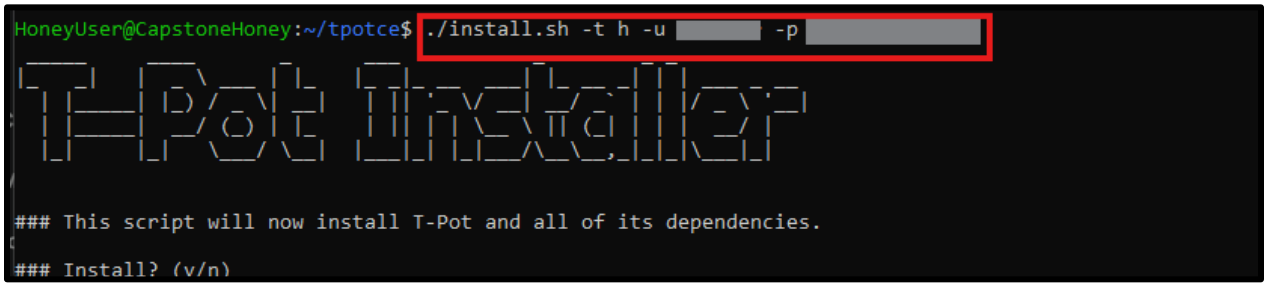


Figure A-13. Accessing the T-Pot web interface via <IP address>:64297

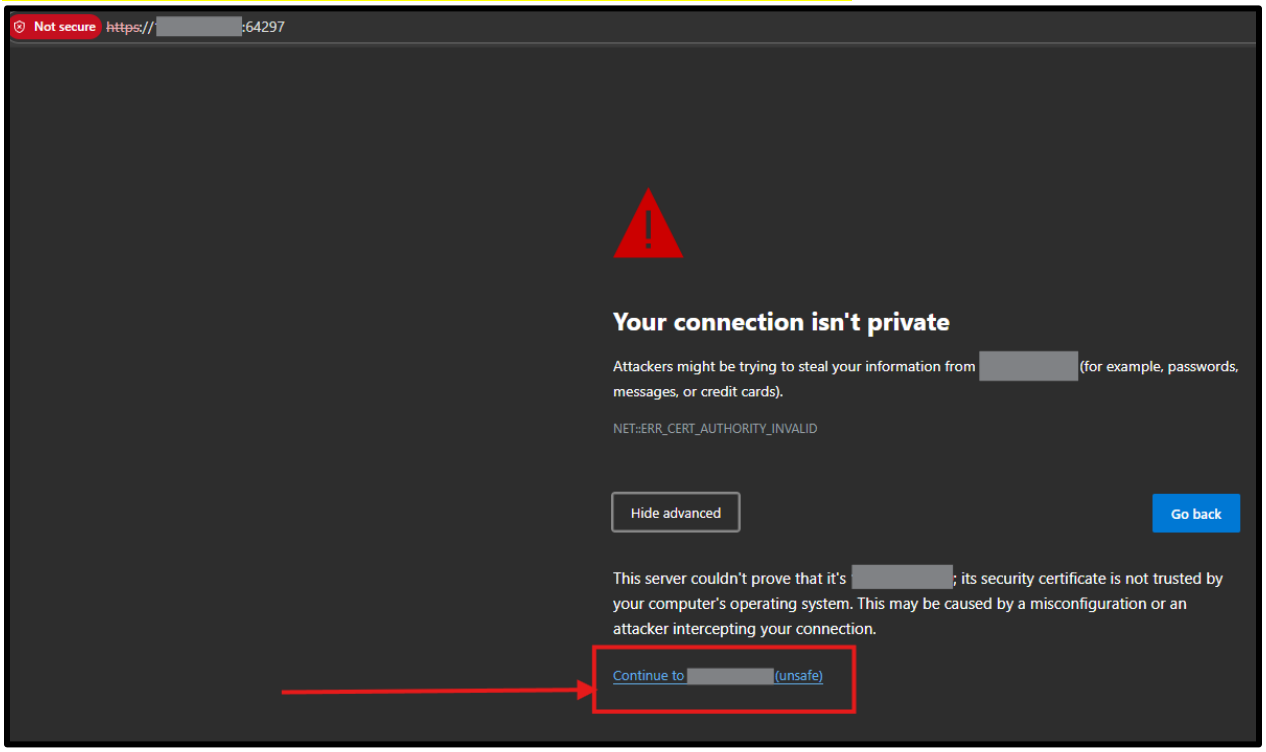


Figure A-14. T-Pot authentication using configured credentials

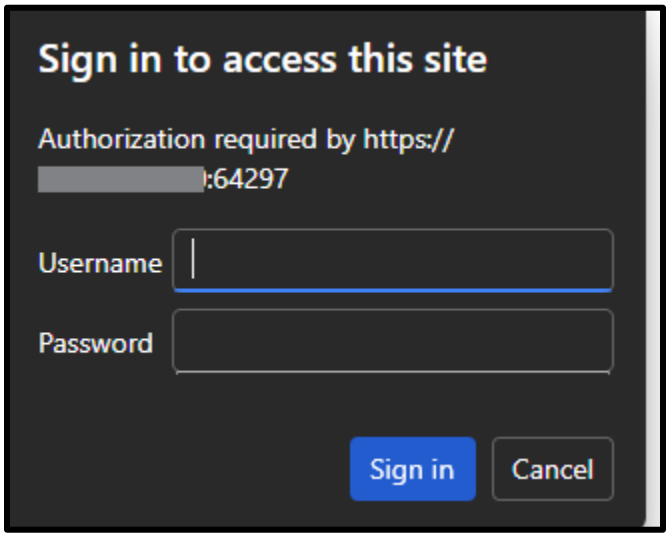


Figure A-15. Successful T-Pot deployment confirmation

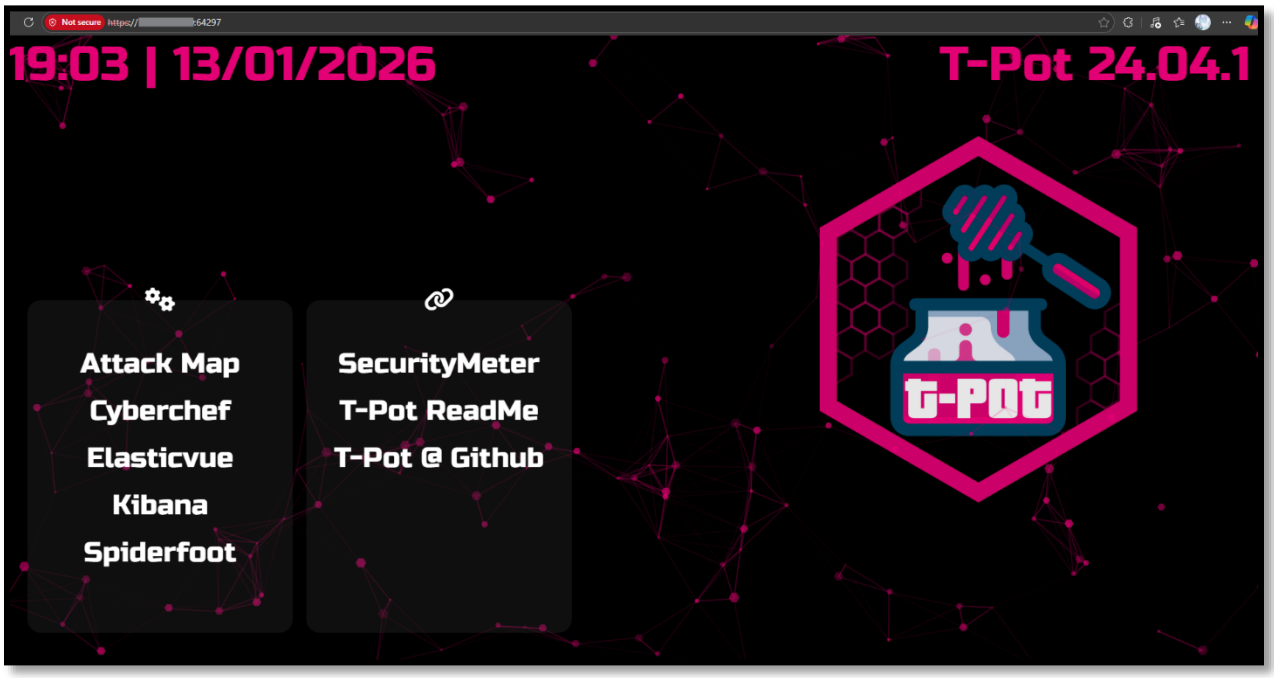
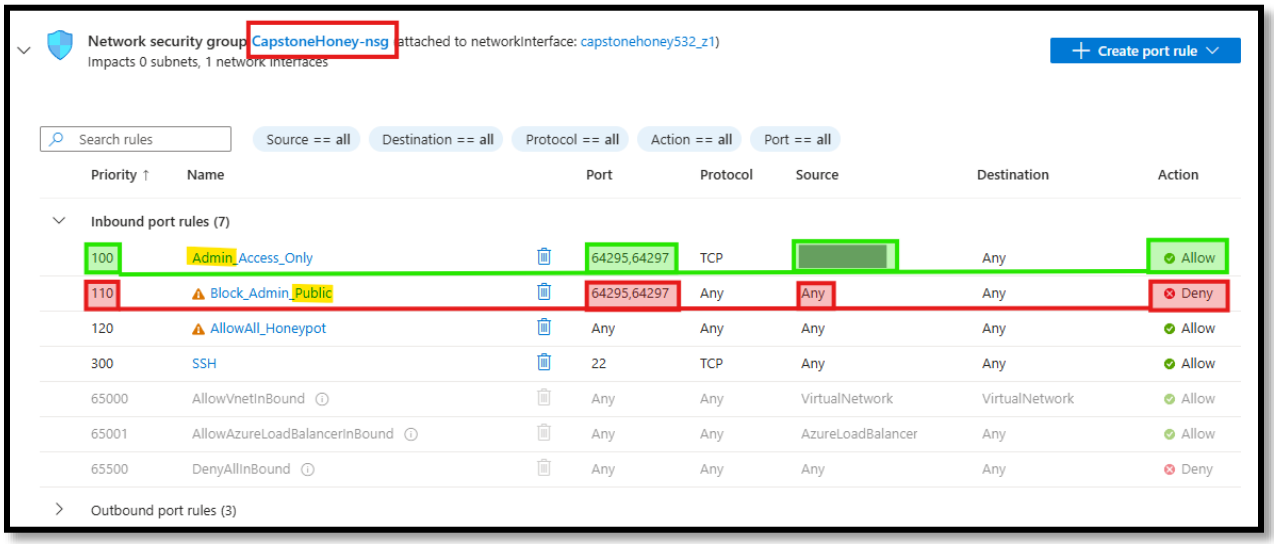


Figure A-16. Inbound Security Rules for Honeypot Management and Exposure



Appendix B

T-Pot Attack Analysis and Data Visualization

Figure B-1. T-Pot integrated attack map with live feed indicating real-time attacks against the honeypot

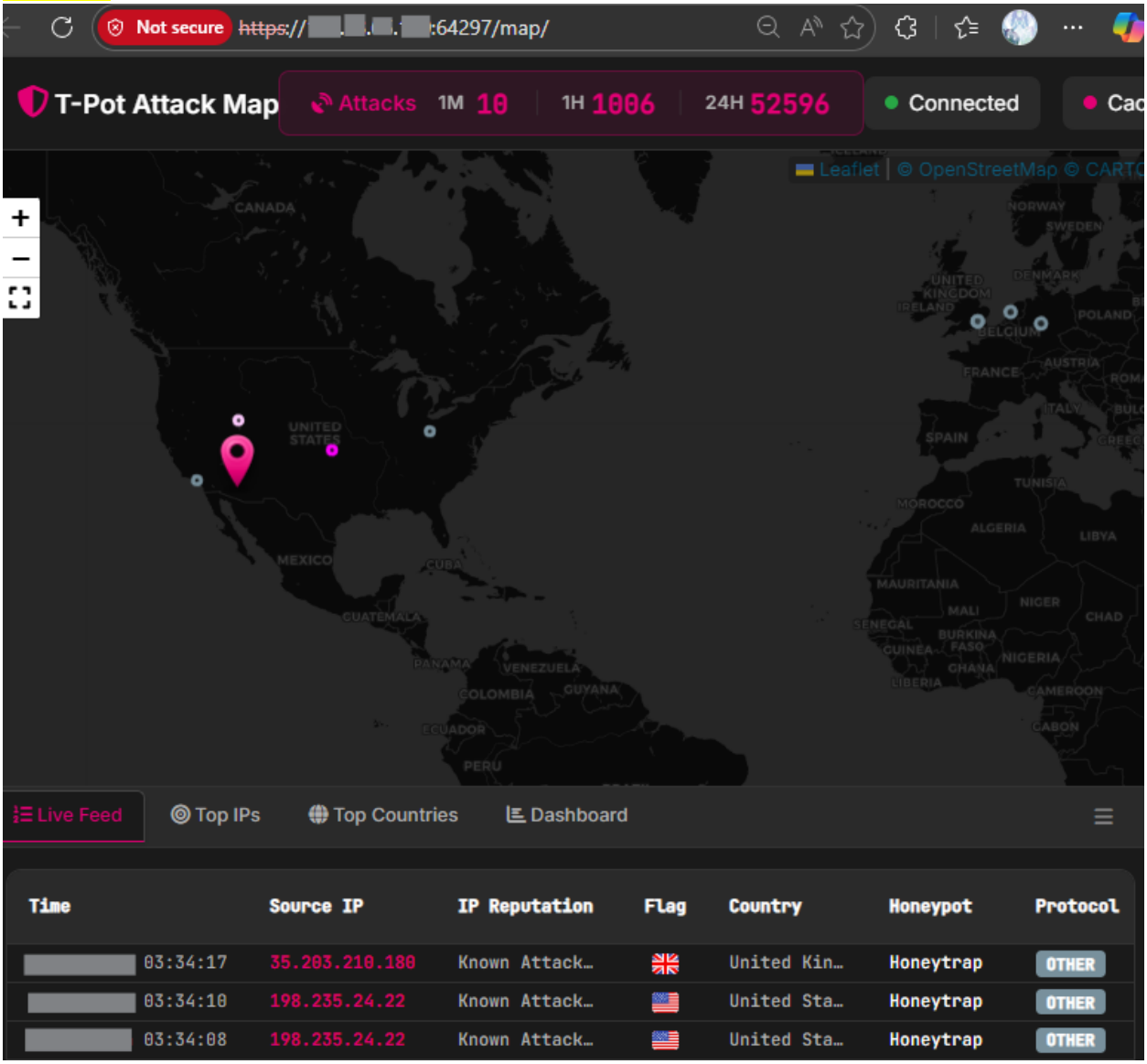


Figure B-2. Kibana dashboard displaying attack data collected over the last seven days, showing 478,000 established connections, including histograms of most targeted destination ports and top

source countries

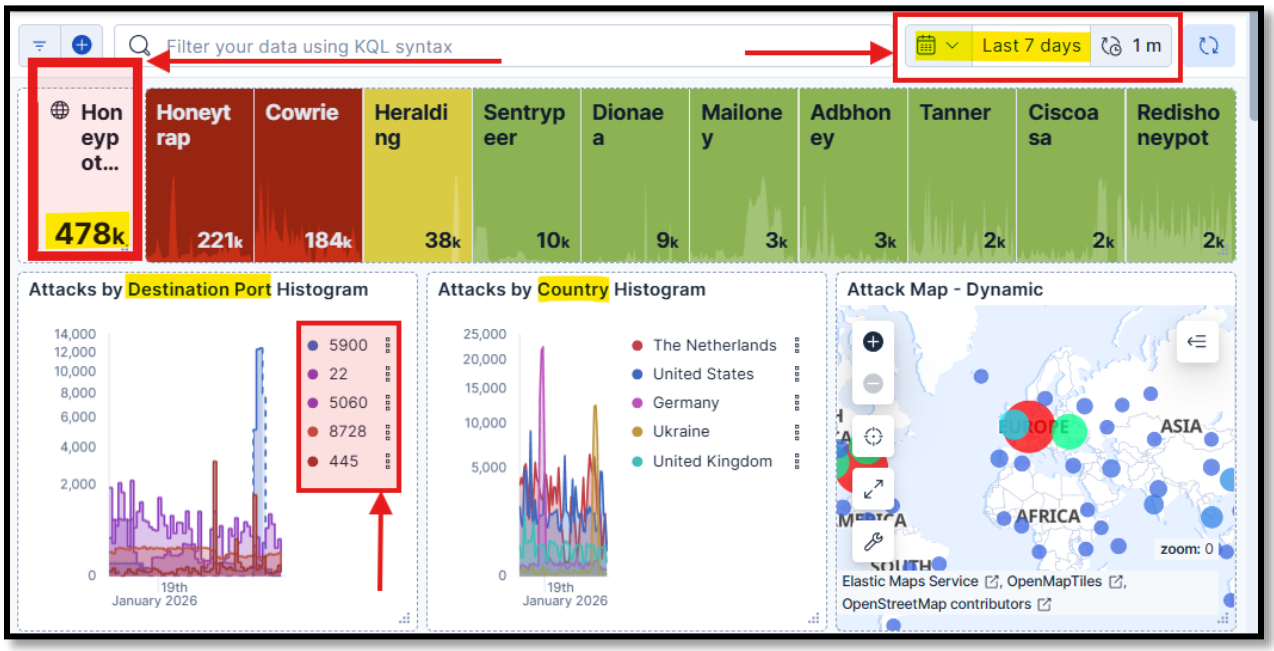


Figure B-3. Kibana filter configured using IP reputation data associated with known attackers, displaying the most frequently targeted ports by identified malicious sources over the last seven days

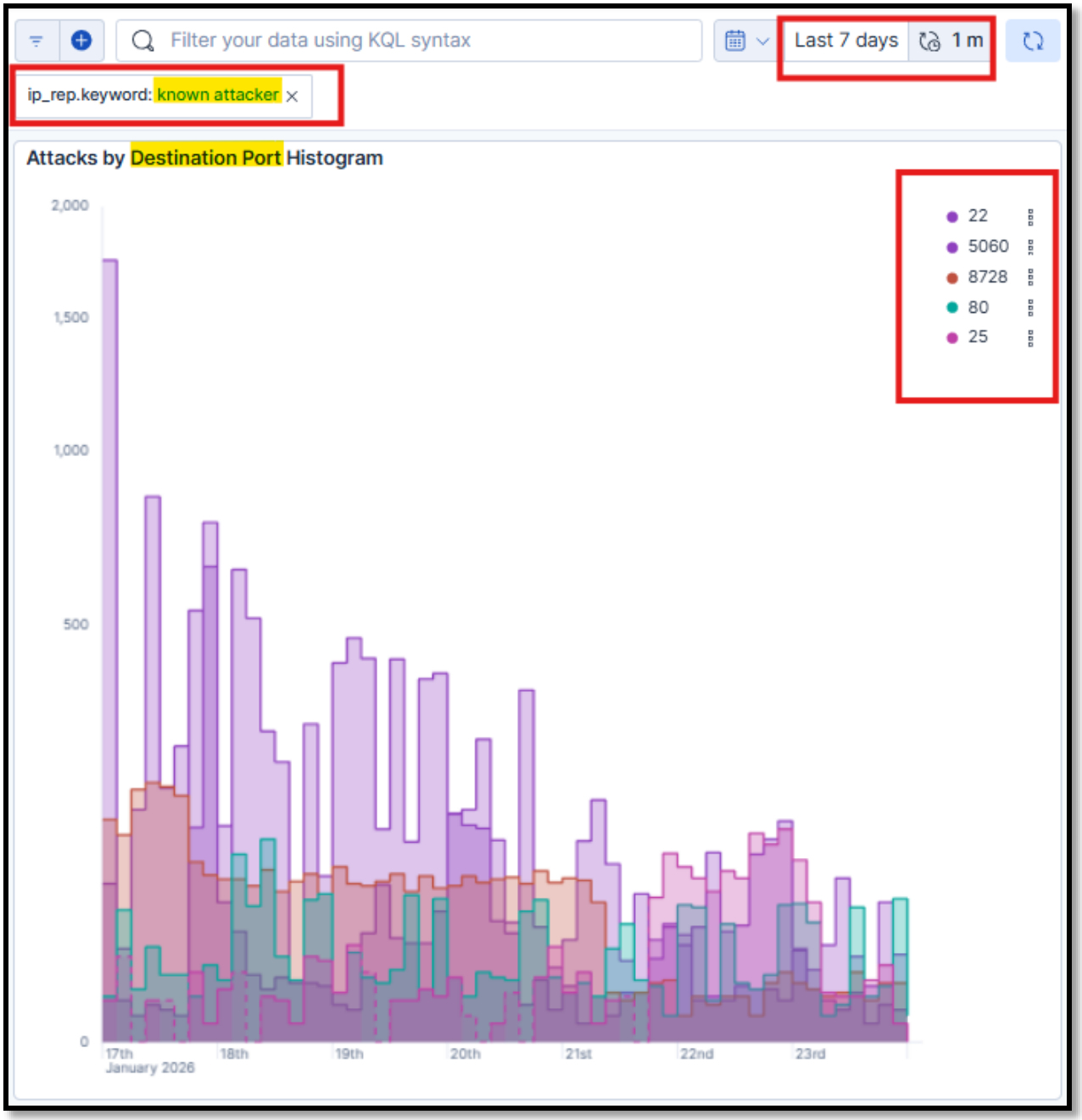
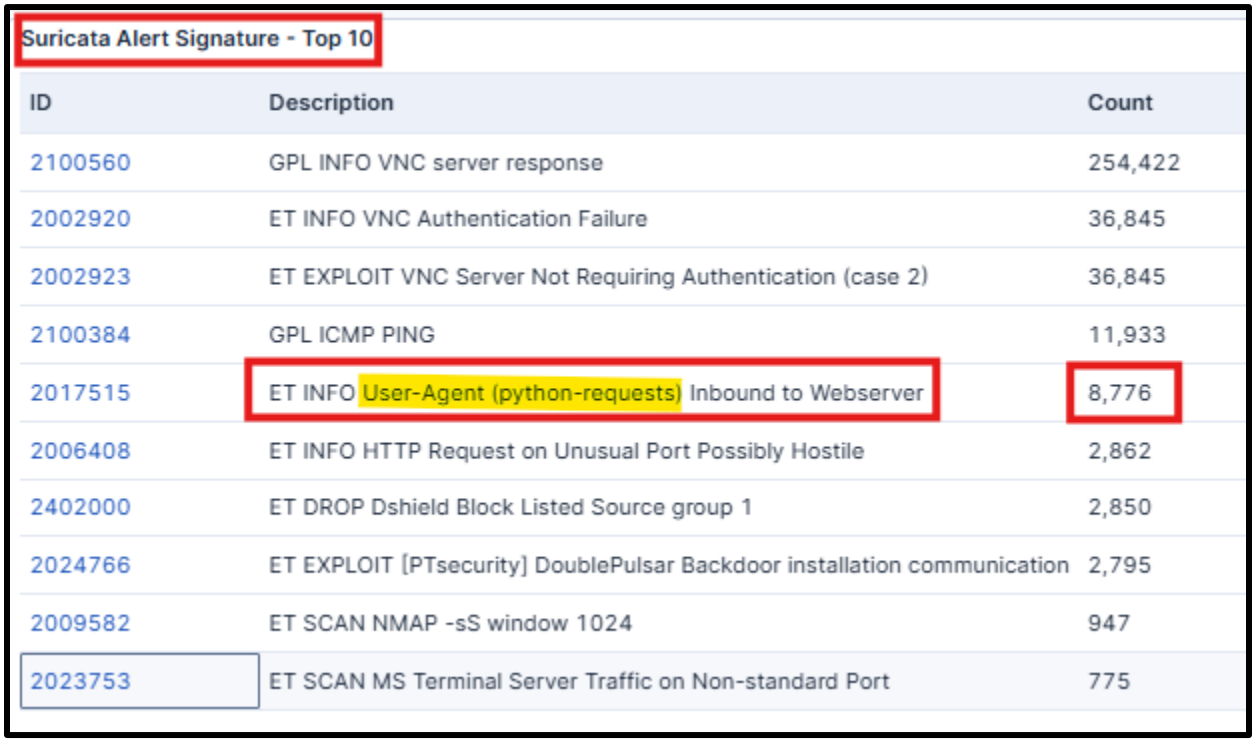


Figure B-4. Suricata alert signature analysis showing one of the top ten signatures related to the user-agent 'python-requests', accounting for 8,776 detected events

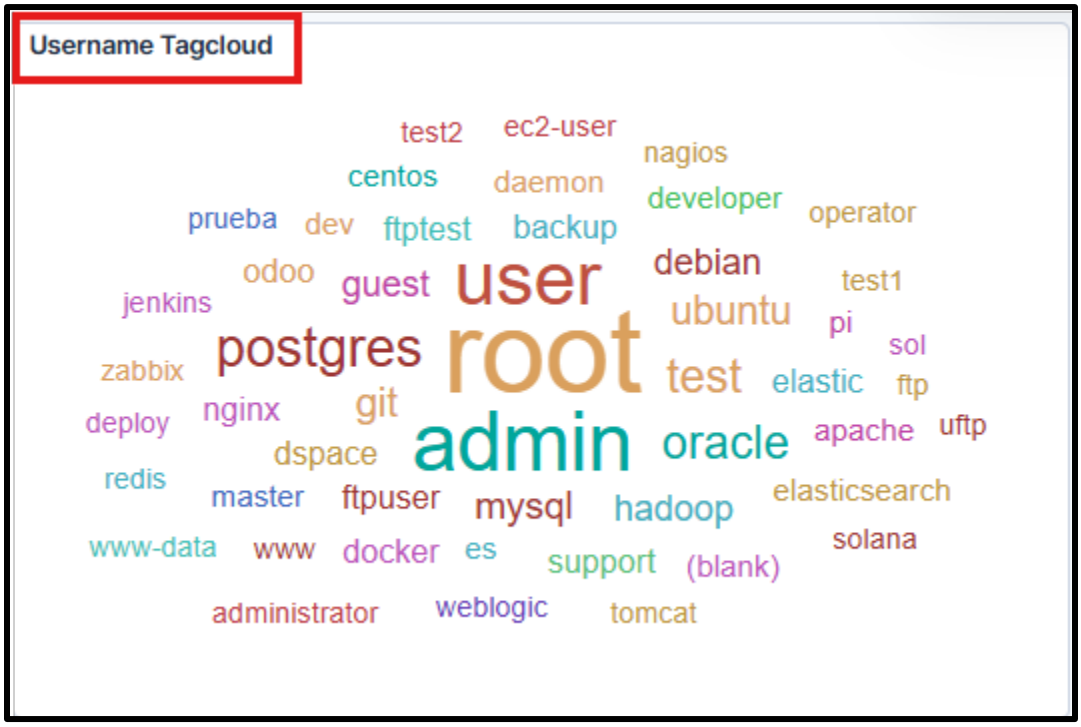


Suricata Alert Signature - Top 10

ID	Description	Count
2100560	GPL INFO VNC server response	254,422
2002920	ET INFO VNC Authentication Failure	36,845
2002923	ET EXPLOIT VNC Server Not Requiring Authentication (case 2)	36,845
2100384	GPL ICMP PING	11,933
2017515	ET INFO User-Agent (python-requests) Inbound to Webserver	8,776
2006408	ET INFO HTTP Request on Unusual Port Possibly Hostile	2,862
2402000	ET DROP Dshield Block Listed Source group 1	2,850
2024766	ET EXPLOIT [PTsecurity] DoublePulsar Backdoor installation communication	2,795
2009582	ET SCAN NMAP -sS window 1024	947
2023753	ET SCAN MS Terminal Server Traffic on Non-standard Port	775

Figure B-5. Visualization of the most common attacker source IP addresses observed targeting the honeypot

Figure B-6. Visualization of the most commonly used usernames observed in authentication-based attacks against the honeypot



Appendix C

Secured Virtual Machine Hardening and Compliance Validation

Figure C-1. Configuration overview of the Secured-VM, including assigned IP address, virtual machine size, and operating system image

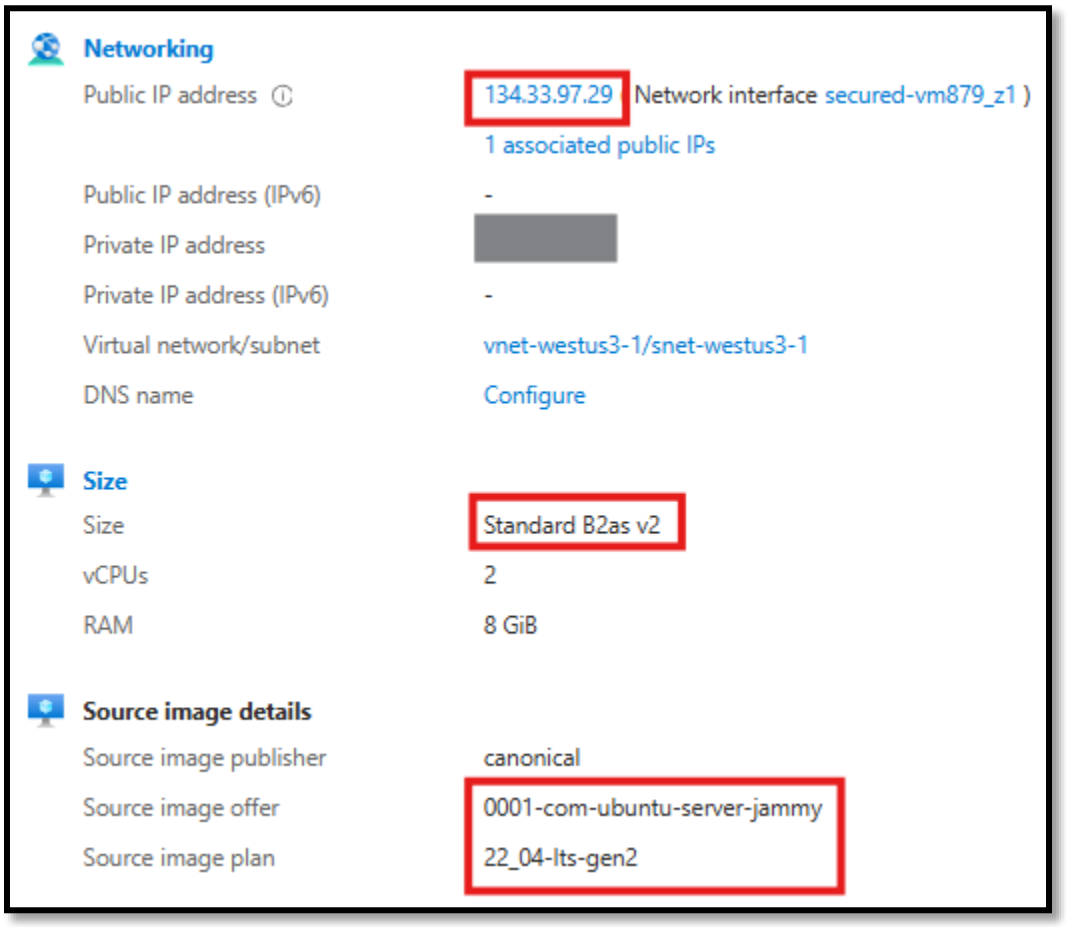


Figure C-2. Web application hosted on the Secured-VM, accessed directly via the virtual machine's public IP address (134.33.97.29)

Figure C-3. Initial OpenSCAP CIS benchmark audit results for the Secured-VM using default Azure-recommended settings, showing a compliance score of 66.377% with five high-severity findings

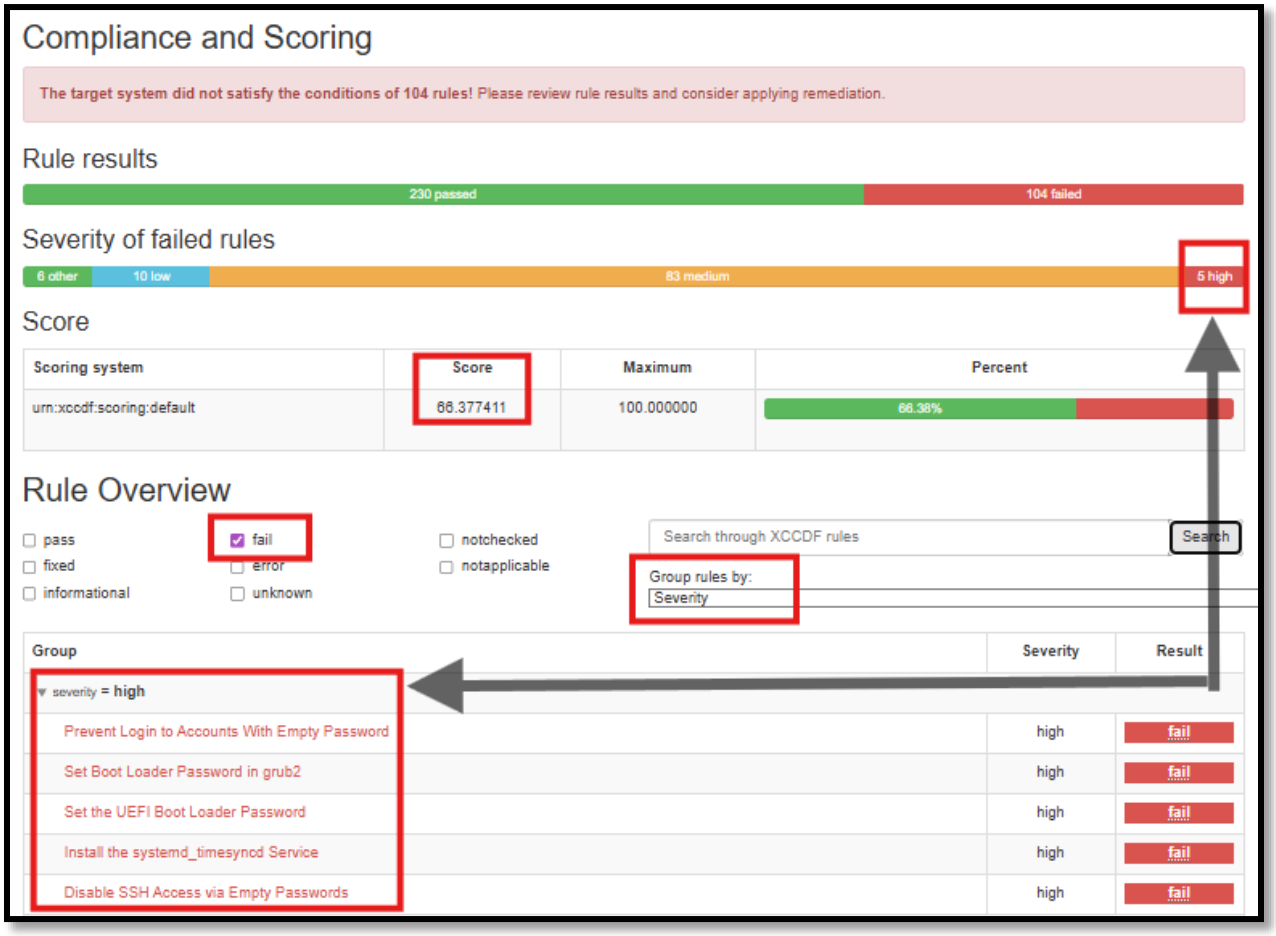


Figure C-4. Modification of the /etc/ssh/sshd_config file on the Secured-VM via Azure Serial Console, showing the PermitRootLogin directive set to "no"

Figure C-5. Modification of the /etc/ssh/sshd_config file on the Secured-VM via Azure Serial Console, showing the PermitEmptyPasswords directive set to "no"

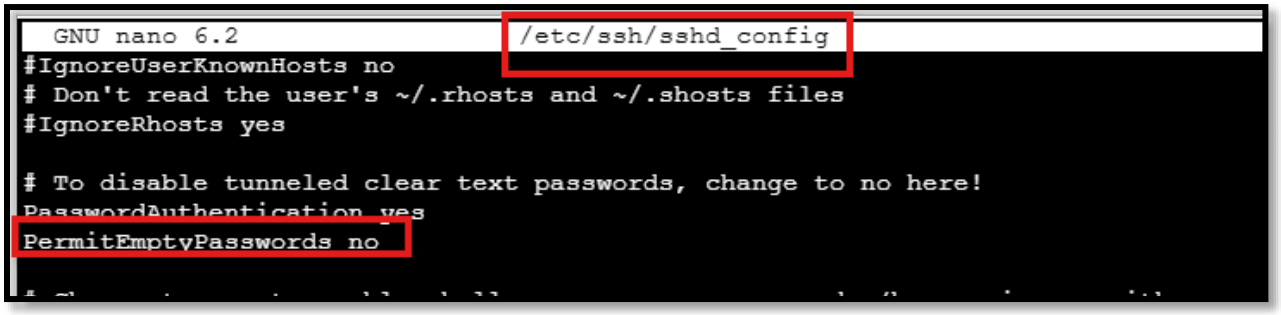


Figure C-6. Follow-up OpenSCAP scan results indicating a reduction to three high-severity findings and an improved compliance score of 68.23% after SSH hardening

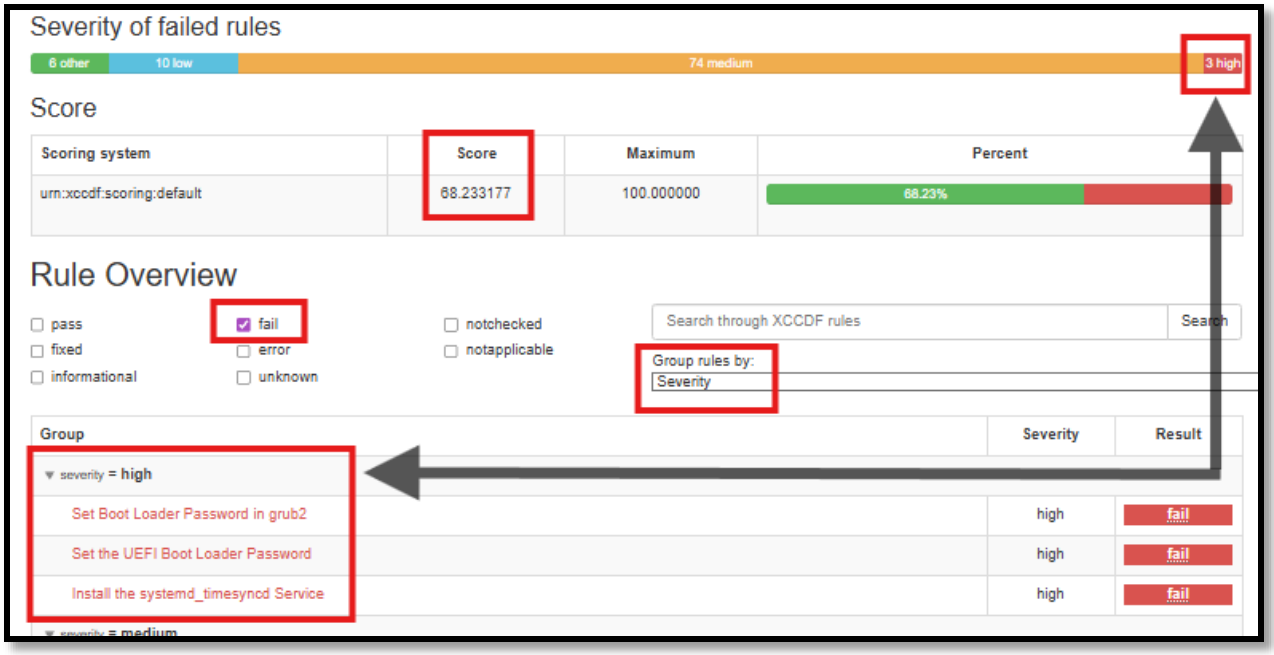


Figure C-7. Implementation of an inbound Azure Network Security Group (NSG) rule on the Secured-VM to block top attacker source IP addresses identified through threat analysis (Figure B-5)

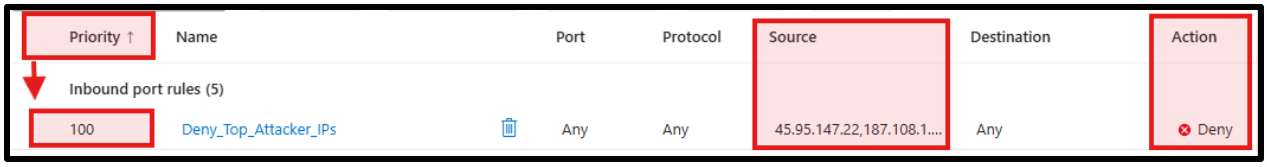


Figure C-8. Implementation of an outbound Azure Network Security Group (NSG) rule on the Secured-VM to restrict suspicious Python-based callback traffic observed during attack analysis

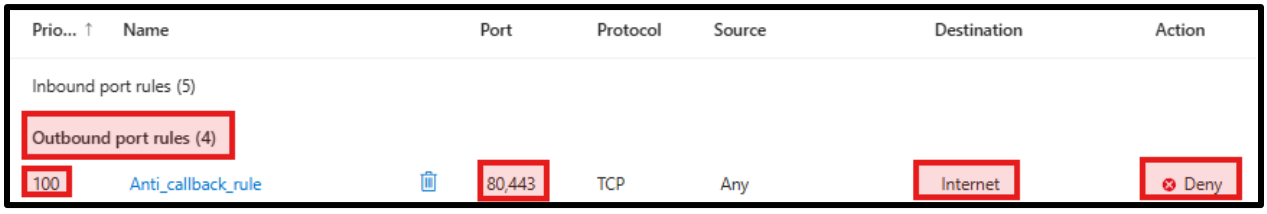


Figure C-9. Final OpenSCAP scan results after applying threat-informed security controls, showing no additional improvement in compliance score, indicating that network-level controls do not directly impact CIS benchmark scoring