Carnegie Mellon University
Software Engineering Institute

Challenge Development Guidelines for Cybersecurity Competitions

Jarrett Booz
Leena Arora
Joseph Vessella
Matt Kaar
Dennis Allen
Josh Hammerstein

Month and Year (date added at time of publication)

TECHNICAL REPORT
CMU/SEI-2022-TR-005
DOI: 10.1184/R1/19597357

CERT Division

http://www.sei.cmu.edu

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

This report was prepared for the SEI Administrative Agent AFLCMC/AZS 5 Eglin Street Hanscom AFB, MA 01731-2100

DM22-0928

Table of Contents

List of Figures

List of Tables

Abstract

Cybersecurity competitions provide a way for participants to learn and develop hands-on technical skills, and they serve to identify and reward talented cybersecurity practitioners. They also form part of a larger, multifaceted effort for ensuring the nation has a highly skilled cybersecurity workforce to secure its critical infrastructure systems and to defend against cyber attacks. To help support these efforts of cultivating the skills of cybersecurity practitioners and of building a workforce to safeguard the nation, this paper draws on the Software Engineering Institute's (SEI) experience developing cybersecurity challenges for the President's Cup Cybersecurity Competition and provides general-purpose guidelines and best practices for developing effective cybersecurity challenges.

1 Introduction

Cybersecurity competitions provide a way for participants to learn and develop hands-on technical skills, and they serve to identify and reward talented cybersecurity practitioners. In addition, cybersecurity competitions form part of a larger, multifaceted effort for ensuring the nation has a highly skilled cybersecurity workforce to secure its critical infrastructure systems and to defend against cyber attacks.

Recently, the United States government highlighted a need for the development of cybersecurity practitioners—a need that cybersecurity competitions can help meet. For example, the Department of Homeland Security (DHS) identified the need to encourage hands-on learning through cybersecurity competitions to address a shortage of skilled cyber defenders [DHS 2021]. Furthermore, in 2019's Executive Order 13870, the president of the United States addressed the need to identify, challenge, and reward the United States government's best cybersecurity practitioners and teams across offensive and defensive cybersecurity disciplines [POTUS 2019]. Well-developed cybersecurity competitions offer a way for government organizations to fulfill that order.

In response to the United States government's call for a federal cybersecurity competition, the Software Engineering Institute (SEI) has been working with the DHS Cybersecurity & Infrastructure Security Agency (CISA) to bring unique cybersecurity challenges to the federal cyber workforce. The SEI has delivered those challenges through an innovative platform as part of the President's Cup Cybersecurity Competition.

This paper draws on the SEI's experience developing cybersecurity challenges for the President's Cup Cybersecurity Competition and builds on lessons learned from years of delivering cybersecurity exercises for the Department of Defense. It provides general-purpose guidelines and best practices for developing effective challenges and addresses the process for planning, developing, and grading cybersecurity challenges; injecting variability into challenges; minimizing cheating and maximizing reusability; testing and reviewing challenges; and, finally, utilizing open source tools to develop challenges and to run cybersecurity competitions.

2 Background

2.1 Cybersecurity Competitions and Challenges

Cybersecurity competitions are exercises geared toward students and professionals in the cybersecurity field. These exercises aim to assess cybersecurity skills, reward experts in the field, and encourage all who participate to learn through hands-on experiences [DHS 2021].

Cybersecurity challenges are the heart of cybersecurity competitions. The challenges provide the hands-on tasks competitors perform as part of the competition. Cybersecurity challenges can take several forms and can involve different activities or responses, such as performing actions on one or many virtual machines (VM), analyzing various types of files or data, writing code, or any other task that cybersecurity professionals may do as part of their job. A single cybersecurity competition might be comprised of several different challenges.

2.2 Targeting Applicable Skills

The goal of cybersecurity challenges is to teach or assess cybersecurity skills through hands-on exercises. With this goal in mind, it is important to design challenges that target skills that are applicable to the cybersecurity workforce.

When building challenges, developers select mission-critical work roles and tasks from the National Initiative for Cybersecurity Education Workforce Framework for Cybersecurity (NICE Framework) [NICCS 2021, Peterson 2020]— a document published by the National Institute of Standards in Technology (NIST) and the National Initiative for Cybersecurity Careers and Studies (NICCS). The NICE Framework serves as the standard for categorizing federal cybersecurity work roles and skills. It defines 52 work roles with detailed information about the specific knowledge, skills, and abilities (KSAs) required to perform tasks in each one. For example, the Cyber Defense Analyst work role contains tasks such as "Perform cyber defense trend analysis and reporting," and other associated KSAs [NICCS 2022].

Using the NICE Framework helps developers focus challenges on critical, in-use skills that best represent the cybersecurity workforce. Each challenge clearly states which NICE work role and tasks it targets. By identifying the knowledge and skills each challenge targets, competitors can easily focus on challenges that address their strengths during the competition and can isolate learning opportunities when using challenges for training.

3 Challenge Planning

Creating successful cybersecurity challenges begins with a comprehensive planning process that involves determining the level of difficulty for each challenge, assessing how many points to award for each challenge, and identifying the tools that are required to solve the challenges.

3.1 Challenge Difficulty

Challenge difficulty is determined by the competition's target audience and the participants' anticipated skill level. Competition organizers want participants to feel engaged and challenged during the competition, which means that challenges cannot be too easy or too hard. Challenges that are too easy will make more advanced participants lose interest, and challenges that are too hard will frustrate competitors. Competitions with participants that have a wide range of skills and experience should include challenges that are suitable for all levels—beginner, intermediate, and advanced.

A single challenge can include a range of difficulties associated with different tasks. Developing challenges with multiple parts is a good strategy to incorporate multiple skill levels in a single challenge. A challenge with multiple parts allows competitors to earn credit for each part as they complete the challenge—rewarding them for progress that has been made.

In contrast, difficult challenges without multiple parts can lead to competitor frustration. Specifically, competitors have a hard time gauging their progress and seeing value for the time they invested on the challenge if there is only a single reward for their effort when the challenge is complete.

3.2 Challenge Points

Points, or the reward granted for successfully solving a challenge, are an important aspect of cybersecurity competitions. Points reward competitors for the time and effort they spend solving each challenge. Moreover, competition organizers can use points to determine competitor placement—competitors with higher scores can advance to future rounds, and organizers can recognize those with the highest points as winners.

The points awarded for solving a challenge, or a part of a challenge, should be commensurate with the difficulty of the challenge and the amount of effort it takes to solve. Beginner-level challenges, or tasks that assess entry-level skills or take a short amount of time to solve, should be worth fewer points than advanced-level challenges, or tasks that assess expert-level skills and take a significant amount of time to solve.

Furthermore, determining the number of points that a challenge, or a challenge part, is worth, and how those points should be distributed between the challenge parts, can be a subjective process. More detail about determining how many points to assign a challenge appears in the Challenge Testing and Review section below.

3.3 Challenge Tooling

Identifying the tools that are required to solve a challenge is an important step of the development process for two reasons. First, it ensures that challenge developers install all required or helpful tools for solving a challenge inside the challenge environment. Second, it is good practice to provide competitors with a list of the tools available in the challenge environment. Making such a list available is important for competitions where organizers provide competitors with the analysis environment so that the competitors are fully aware of all the tools available to them.

Additionally, developers should build challenges that do not require the use of paid or licensed software to be solved. Leveraging open source or free tools, applications, and operating systems is vital because some competitors might not have access to certain software licenses, which would put them at a disadvantage in the competition or even prevent them from completing the challenge altogether.

4 Challenge Development

Creating cybersecurity challenges is both a technical and a creative undertaking. Developers need to be well-versed in cybersecurity subject matter and devise innovative ways to test the skills of the competitors. Challenge development involves identifying the skills a challenge will target and the scenario it will simulate; developing the technical aspects of the challenge; implementing a grading mechanism that is automated and auditable; incorporating variability into challenge deployments; and writing supporting documentation for both internal challenge testers and competitors.

4.1 Pre-Development

Before creating the technical portions of a challenge, developers should begin by identifying the work roles and skills their challenge aims to assess, as described in the Targeting Applicable Skills section above. By identifying the targeted work roles early in the development process, developers can build more precise challenges and avoid including tasks that do not assess applicable skills or that test too wide an array of skills. Challenges that assess a targeted range of skills are better than challenges that assess too many unrelated skills. Once developers define the work role for a challenge, they can move forward with forming a challenge idea.

The challenge idea includes the technical tasks that competitors will complete and the location where the events of the challenge scenario take place. The technical tasks that comprise the challenge can be unique to it, can be modifications of a previous challenge, or can be replications or simulations of a real-world cybersecurity event. All challenge tasks should resemble the kind that professionals complete as part of their job. If developers borrow or reuse the technical components of a previous challenge, they must make modifications that uniquely distinguish the new challenge from the original. Such modifications are critical to ensure that competitors who completed the original challenge do not recognize the new one in a way that might give them an unfair advantage for completing it.

At a minimum, challenges should include a scenario that describes the VMs included in the challenge and the tasks the competitor must complete. It is important for competitors to clearly understand the operating environment and the tasks at hand so they can focus their time and energy on solving the technical problem in the challenge. Some challenges include a detailed creative scenario that places competitors in real-world roles, such as an IT specialist, incident responder, penetration tester, or others. Developers are free to be as creative as they wish when building the scenario. Topical challenges based on real-world cybersecurity events offer another way to add unique and creative scenarios to challenges.

4.2 Challenge Development

Developers begin creating the technical aspects of the challenge only after they have identified the skill it will target and the scenario it will simulate.

The technical components of challenge development generally involve VM, network, and service configuration. The configuration developers implement as part of challenge development ensures that the challenge environment deploys correctly when competitors attempt the challenge.

Challenge development might include configuring VMs or services to incorporate known vulnerabilities; configuring routers, firewalls, services, etc., to the state developers want; staging attack artifacts or evidence throughout networks or logs; and other actions that prepare the environment for the challenge. Additionally, challenge developers can purposefully misconfigure aspects of the environment if the challenge targets skills in finding and fixing misconfigurations.

4.3 Best Practices for Developing Challenges

Because each challenge targets different skills, there is no standard process for developing a cybersecurity challenge. However, developers should follow these best practices:

- Ensure that the technical skills assessed by the challenge are applicable in the real world.
- Ensure that the tools required to solve the challenge are free to use and available to the competitors.
- Make a list of the tools that are available in the hosted environment available to competitors.
- Ensure that challenges do not force competitors down a single solution path. Competitors should be able to solve challenges in any realistic manner.
- Remove unnecessary hints or shortcuts from the challenge, including removing command history, clearing browsing data, and removing other data that could give hints or give competitors a shortcut to solving the challenge.

4.4 Challenge Grading

Grading, or how developers determine competitors' success in a challenge, is vital to providing a fair, consistent, and enjoyable experience for challenge competitors. In general, developers should automate grading through an authoritative server that is responsible for receiving answer submissions from the competitors and determining how many points to award for each submission.

Submissions for all challenges should be automatically reviewed to ensure competitors have submitted the correct answer. The submission system should generally ignore differences in capitalization, white space, special characters, and other variations that are ultimately irrelevant with respect to whether the submission is right or wrong. For example, if a challenge requires the submission of a file path, the system should allow competitors to enter either "/" or "\" as path separators without impacting answer correctness. Ignoring such formatting inconsistencies ensures that developers build a system that does not penalize competitors for submitting items with variations that do not fundamentally change the correctness of their answer.

Ignoring variations when grading submissions might seem contradictory to assessments or operational readiness evaluations where exact precision is required. However, cybersecurity competitions have goals and considerations beyond evaluating operational proficiency, such as ensuring a fair competition and encouraging broad participation. As a result, it is better to provide some flexibility with the grading—so long as the answer is fundamentally correct—rather than penalizing competitors based on a technicality with the formatting.

The system can grade challenges using several different methods, such as token discovery, question-and-answer problems, or environment verification. The sections that follow discuss these methods.

4.4.1 Token Discovery

In token discovery grading, competitors must find a string or token that follows a defined format (these tokens can also be called "flags"). Developers can place the token in any part of the challenge where the competitor will find it by completing the challenge tasks.

Example of Token Discovery

A challenge requires competitors to access a website on a non-standard port or protocol. Once competitors navigate to the website, they can download a file that contains the 32-character submission token, as shown in Figure 1 below.

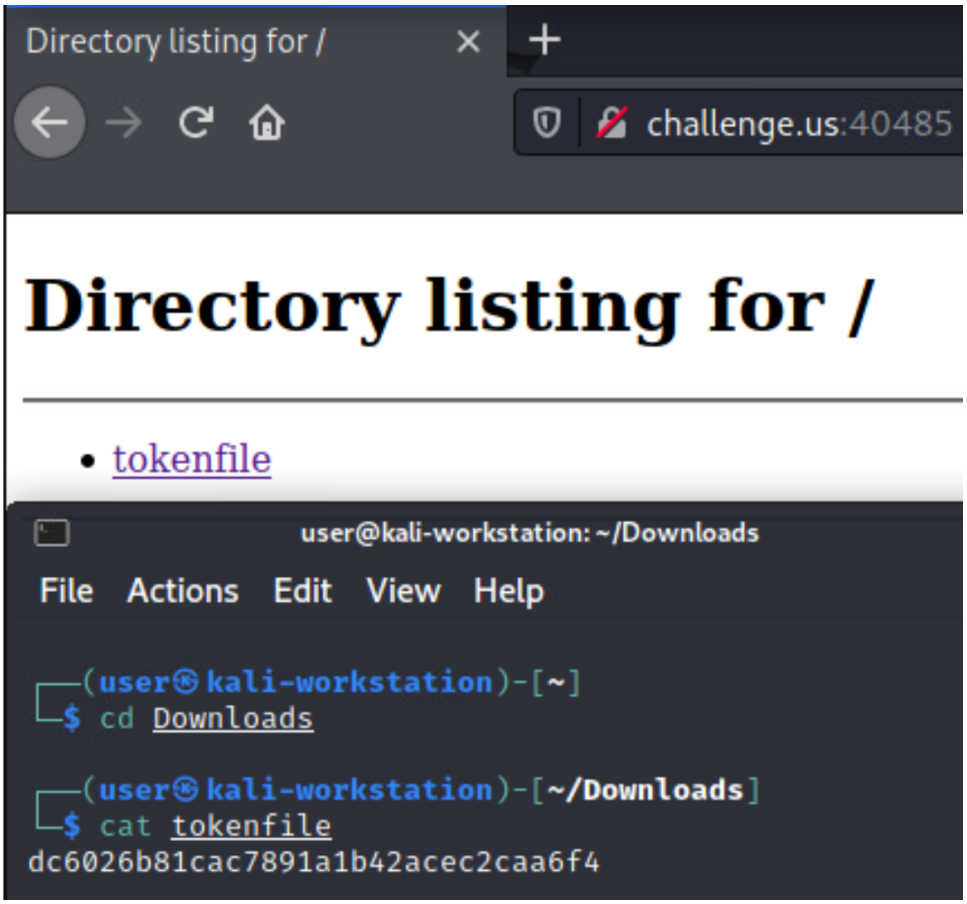


Figure 1: Token Discovery Grading

Here are some best practices for using token discovery grading:

- The format of the token should be well defined and easy to type or copy and paste (e.g., known-length hexadecimal strings). Clear formatting ensures that it will be obvious to competitors when they have found the correct token, and it will be easy for competitors to submit it.

- Challenges should not contain incorrect strings that match the submission format (i.e., false flags). False flags can lead to frustration because competitors do not know whether they have found the correct token, or whether they have found an incorrect token that meets the expected criteria.

The following challenges are good use cases for token discovery grading:

- challenges where finding a file proves that the competitor completed the required tasks
- challenges where finding a string in network traffic, files, or other data sources proves that the competitor completed the required tasks

4.4.2 Question-and-Answer Problems

With question-and-answer problems, the competitor must find the correct answer to one or more questions by performing challenge tasks. The answers to the challenge questions can take several forms, such as entering file paths, IP addresses, hostnames, usernames, or other fields and formats that are clearly defined.

Example of a Question-and-Answer Problem

A challenge requires competitors to respond to an attack on a domain controller. The attackers have used the popular Golden Ticket attack to forge a Kerberos ticket that gives arbitrary users access to domain resources. The challenge asks the competitor to provide the username that the attacker is using with the Golden Ticket.

The question states, *"Provide the username of the user utilizing the golden ticket (i.e. Firstname.Lastname)."*

The competitor searches the system event log for evidence of Golden Ticket use and discovers the username "Alexander.Hamilton" as the answer to the challenge question, as shown in Figure 2.

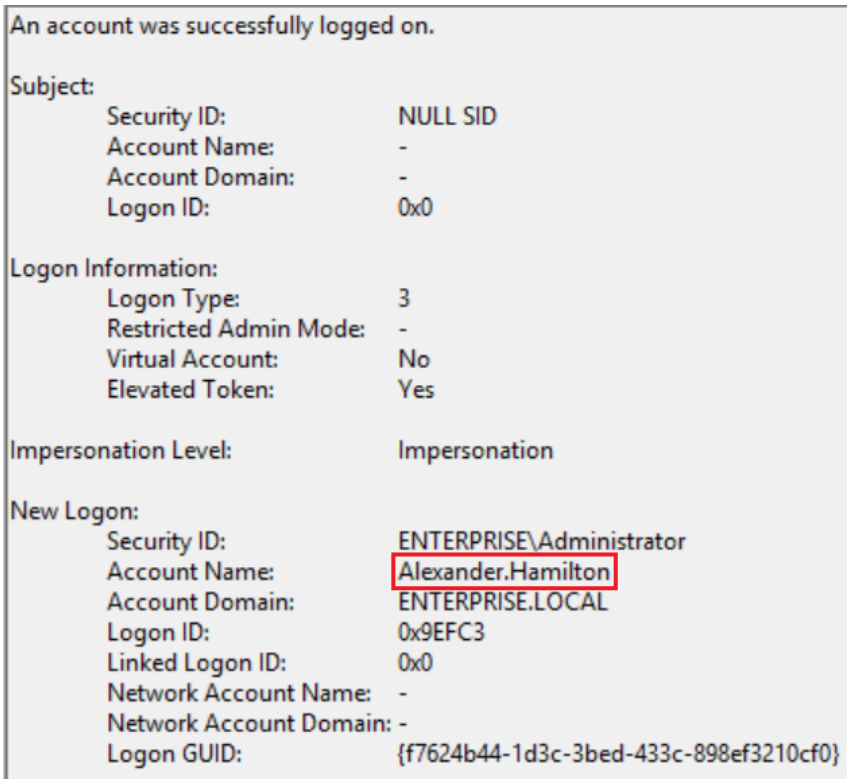


Figure 2: Question-and-Answer Grading

Here are some best practices for question-and-answer grading:

- The questions the challenge asks should be specific, such that there is only one correct answer.
- The format of the expected submission should be clearly defined and accompanied by an example so competitors know how to submit their answer.

A good use case for question-and-answer problems are challenges where providing a hostname, username, IP address, file path, or other unique, identifying piece of information proves that the competitor completed the required tasks.

4.4.3 Environment Verification

In environment verification grading, the system grades competitors based on changes they make to the challenge environment. Challenges can task competitors with fixing a misconfiguration, mitigating a vulnerability, attacking a service, or any other activity where success can be measured dynamically. Developers can use an authoritative grading system or service colocated within the competitor's challenge environment to assess the state of the environment and verify that competitors have made all required changes. The grading system will run a program or script that uses remote access tools (e.g., SSH, PsExec, etc.) or performs an action over the network to verify that the competitor has completed the required tasks. When the grading system verifies changes to the environment state, it provides competitors with a success token.

Example of Environment Verification

A challenge requires competitors to mitigate a malware infection. The competitors must stop the malicious process from running (Grading Check 1) and remove the malicious file from the file system (Grading Check 2).

To check their progress and receive success tokens for each grading check, competitors visit the in-game grading website. Figure 3 below shows the landing page for the grading site.

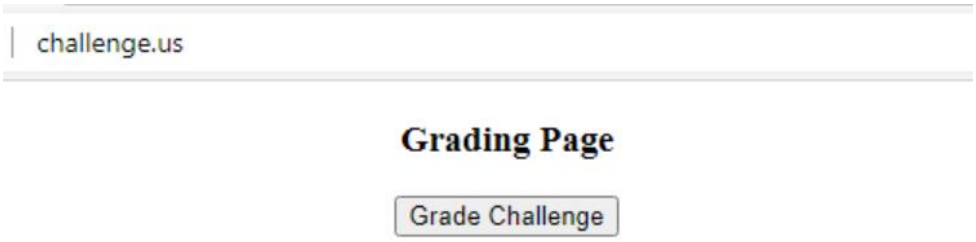


Figure 3: Environment Verification Grading

When they navigate to the in-game grading website, competitors can click the "Grade Challenge" button to initiate the grading script. The grading script accesses the infected system remotely and follows the workflow shown in Figure 4.

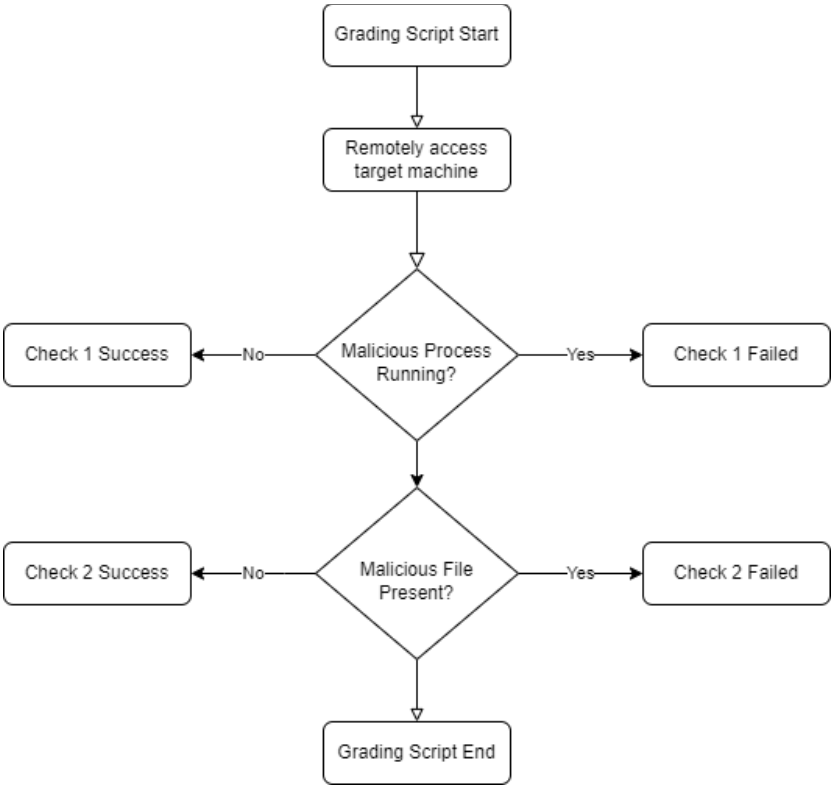


Figure 4: Example Grading Script Workflow

After the script executes, the grading server displays the results and success tokens to the competitor, as shown in Figure 5.

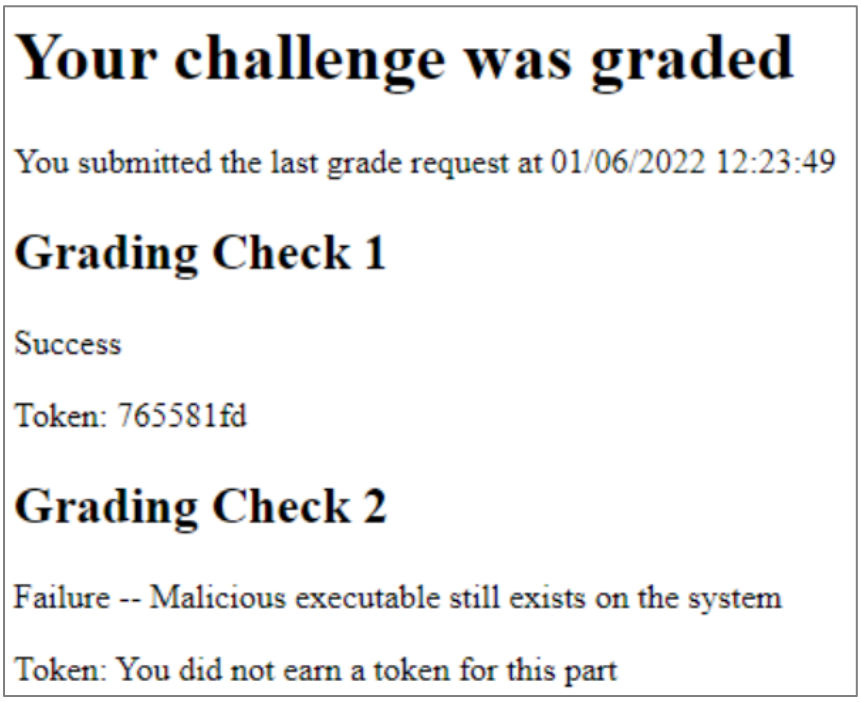


Figure 5: Example Grading Checks

Competitors can continue altering the challenge VMs until they perform the actions required to pass Grading Check 2.

Competitors can submit the success token manually or the grading system can submit the token on the competitor's behalf for automatic scoring updates. Success tokens in environment verification grading are similar to those in token discovery grading. However, with environment verification, competitors do not have to discover the token. Instead, the system provides them with the token when their environment meets the success conditions.

Here are some best practices for environment verification grading:

- The challenge should tell competitors which parts of the environment the system is assessing. The challenge should clearly define all grading requirements and success conditions.
- The grading system should provide feedback to competitors. Doing so informs them that grading has occurred so that they are aware that the grading system is functioning correctly. In addition, providing feedback about failed verification checks gives competitors guidance to understand their mistakes and where to look when adjusting their environment for subsequent grading tasks.
- Developers should engineer grading scripts to withstand failure and to implement detailed logging. Grading scripts should handle programming exceptions in a way that prevents unexpected failure, and they should also log all actions and their results so that challenge organizers can quickly address issues and competitor questions by viewing log output.

Good use cases for environment verification grading include the following:

- challenges where competitors must configure network devices, hosts, services, etc., in a way that the system can verify dynamically to determine whether the competitor completed the required tasks
- challenges where competitors must create a condition (e.g., establish a network connection, create a file, create a user, etc.) that the system can dynamically verify to determine whether the competitor completed the required tasks

4.5 Challenge Variation

Including some level of variation between different deployments of a challenge allows there to be different correct answers for the same challenge, which is important for two reasons. First, it helps promote a fair competition by discouraging competitors from sharing answers. Having different answers for each user in the same challenge mitigates the risk of competitors sharing challenge answers and earning points for a challenge that they did not solve themselves. Second, it allows competition organizers to reuse challenges without losing educational value. Challenges that can be completed numerous times without resulting in the same answer enable competitors to learn and hone their skills through repeated practice of the same challenge.

There are several ways developers can introduce variation into challenges depending on the type of grading that they use. The sections below explain different approaches for introducing variation.

4.5.1 Token-Based Variation

Challenges using token discovery or environment verification grading can randomly generate unique tokens for each competitor when the challenge is deployed. Developers can insert dynamically generated submission tokens into the challenge environment (e.g., inserting guestinfo variables into VMs), and they can copy them to the locations where they expect competitors to receive the challenge answers.

Example of Token-Based Variation

A challenge requires competitors to exploit a service. Once they acquire access to the remote service, competitors will find a file called **token.txt**.

When competitors launch the challenge, the system generates random tokens and passes them into the target VM. The target VM has a script that writes the generated token into the **token.txt** file. Figure 6 below outlines this process.

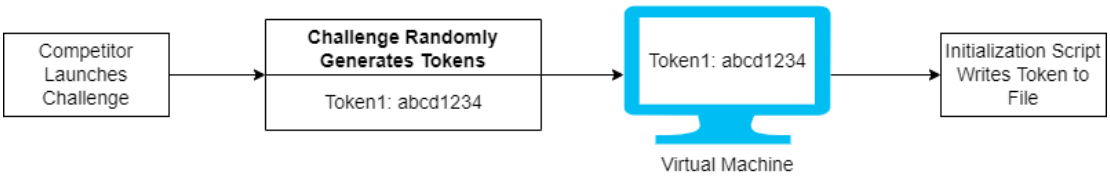


Figure 6: Randomly Generated Token Workflow

4.5.2 Question-and-Answer Variation

In question-and-answer challenges, developers can introduce variation by configuring different answers to the same questions or by asking different questions. The sections below address different ways to do so.

Question-and-Answer Variation Using Initialization Scripts

To create different answers to the same questions, the challenge can use an initialization script to configure the VMs differently based on the challenge parameters.

Example of Question-and-Answer Variation Using Initialization Scripts

A challenge requires competitors to examine a VM that has been infected with malware. Developers can build the challenge using an initialization script that changes the name of the malicious file. The challenge asks all competitors the same question about the malicious file name, but the answers will be different due to the fact that the initialization script will change the file name. Figure 7 below outlines this process.

Figure 7: Randomly Generated Question-and-Answer Workflow

Question-and-Answer Variation Using Challenge Variants

When dynamic configuration of a challenge via an initialization script is not possible, developers can create several versions of a challenge, which get randomly deployed to competitors. Developers preconfigure VMs for different challenge variants to have different correct answers to the challenge questions.

Example of Question-and-Answer Variation Using Challenge Variants

A challenge requires competitors to respond to an attack. The challenge question asks competitors to identify the subnet that the attacker is sending malware from. This challenge has four variants—developers configure the variants such that the attacker is using a different subnet in each one.

When competitors launch the challenge, a random variant is deployed. The answers that competitors find correspond to the variant deployed for their challenge. Figure 8 below outlines this process.

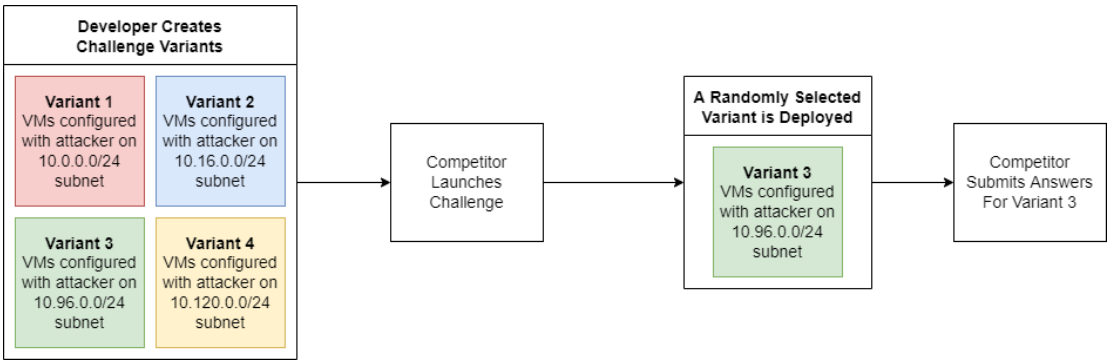


Figure 8: Random Variant Deployment Workflow

Question-and-Answer Variation Using Different Questions

Alternatively, asking different questions about the same environment is a way to introduce variation into a challenge without any additional development (e.g., initialization scripts or variant-specific VMs). To ensure fairness, it is vital that the different question sets are equally difficult and time consuming to solve.

Example of Question-and-Answer Variation Using Different Questions

A challenge requires competitors to examine a packet capture file (PCAP). All competitors are given the same PCAP file. To introduce variation, competitors are asked different questions about the PCAP.

The system randomly chooses the question each competitor answers from the following set of four questions:

1. Which IP address sent the most packets?
2. Which IP address received the most packets?
3. Which IP address sent the most bytes?
4. Which IP address received the most bytes?

Each of these questions are equal in their difficulty and in the amount of time they take to solve because competitors can answer all of them by looking at the endpoint statistics menu in Wireshark, shown in Figure 9 below.

Ethernet · 8 | IPv4 · 8 | IPv6 | TCP · 8 | UDP

Address	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes
10.5.5.200	6	324	5	282
14.93.78.40	4	1,162	5	270
19.18.17.16	340	18k	538	32k
40.102.198.185	5	282	6	324
47.16.19.220	337	2,004k	338	18k
128.131.6.181	538	32k	340	18k
139.189.0.108	5	270	4	1,162
237.37.170.11	338	18k	337	2,004k

Figure 9: Wireshark Statistics Showing Randomly Generated PCAP

4.6 Challenge Documentation

As part of developing a challenge, developers must write supporting documentation. The challenge guide and the solution guide are two pieces of critical documentation that allow developers to test the challenge and competitors to solve it.

4.6.1 Challenge Guide

The challenge guide is the document that accompanies the challenge in the gameboard environment. It is visible to the competitors at the time they begin the challenge. The challenge guide provides a short description of the challenge, the skills and tasks the challenge assesses, the scenario and any background information that is required to understand the environment, machine credentials, and the submission area or areas.

Example of Challenge Guide Documentation

This section provides only a high-level outline of the sections contained in a sample challenge guide and the information developers should include in each section. A discussion of each section of the challenge guide follows the short outline.

Introduction

- brief challenge description
- list of assessed skills and tasks

Background

- short, one-to-three sentence introduction to the scenario and context of the challenge

Getting Started

- description of how the competitors should get started
- explanation of what competitors need to accomplish within the challenge
- explanations of any other critical information and instructions

Credentials

- a table with a row dedicated to each VM or service and the corresponding username and password to access them

Submission

- descriptions of the type of grading that the challenge uses
- list of submission area or areas with example submissions to clarify the information competitors must submit

The short description and the list of skills assessed by the challenge give competitors context about the challenge before they attempt to solve it.

The challenge scenario and background information give competitors a clear call to action for the challenge. Understanding the scenario, the required tasks, and where to begin is vital for success.

If competitors do not know what they must do or do not have a valid starting point, the challenge may be unsolvable. Including all credentials as part of the challenge documentation is required to ensure competitors can access all VMs, tools, etc., that are part of the challenge.

Finally, the guide includes the submission area to reinforce the challenge tasks and to provide a location for competitors to submit their answers.

4.6.2 Best Practices for Writing Challenge Guides

The following are best practices for writing a challenge guide:

- The challenge document should describe the scenario in a way that competitors can easily follow and understand. Competitors may not know how to begin solving a challenge if the document is unclear.
- The challenge scenario and background information should avoid logical leaps. Leaps in logic make the scenario hard to understand for competitors who do not immediately connect the dots the way the author intended.
- The challenge difficulty should not hinge on logical leaps or on information intentionally left out of the guide.

Appendix A provides an example challenge guide from a 2021 President's Cup challenge.

4.6.3 Solution Guide

The solution guide is another vital piece of documentation that developers create as part of challenge development. The solution guide provides a walk-through of one way to complete the challenge (there are usually other ways to complete the challenge that the solution guide does not cover). During testing, developers use the solution guide to ensure that the challenge is possible to solve. Developers can also release the solution guide to the public after the conclusion of the competition to serve as a community learning resource.

The intended audience for this guide is the general cybersecurity community. As such, developers should write the solution guide with the assumption that the reader is familiar with basic IT and cybersecurity skills but is not an expert in the field. The guide clearly and thoroughly leads the reader through a full solution of the challenge. Screenshots and other images can often be helpful additions to these guides.

Appendix B provides an example solution guide from a 2021 President's Cup challenge.

4.7 Challenge Testing and Review

After developers build a challenge, it should go through several rounds of testing and review. Developers test challenges to ensure quality, and they review them to estimate the challenge's difficulty.

4.7.1 Quality Assurance Testing

After developers build a challenge, they should test it thoroughly. Developers perform an initial round of testing to catch any errors that arise during the challenge deployment and initialization process. Developers also ensure that competitors can fully solve the challenge in at least one way.

After developers complete the first round of testing and mark their challenge as ready for further testing, the challenge should be tested several times by qualified technical staff who are unfamiliar with the challenge. Using testers who are unfamiliar with the challenge is the most accurate way to realistically simulate how the challenge—which competitors will try to solve without also ever having seen it before—will play out. Testers should be encouraged to attempt solving the challenge on their own but may be provided the developer's solution guide for help.

The testers should ensure each challenge passes the following quality assurance tests:

1. The challenge deploys as expected and without errors.
2. The challenge VMs are accessible (hidden VMs should be hidden from the user interface but deployed).
3. The challenge is solvable (this item should include making sure VMs have the required tools, and that the system awards points for correct submissions).
4. There are no unintentional shortcuts to solving the challenge (e.g., command history left behind, default credentials, etc.).
5. Challenge instructions and questions are properly formatted and give a clear indication of what competitors need to do.

4.7.2 Challenge Review

Challenge testers should also take notes about the content of the challenge as part of the review process. The notes should include estimates of how difficult the challenge is and how long the tester believes it would take competitors to solve.

Testers can measure challenge difficulty in two ways: by indicating the experience level required (beginner, intermediate, advanced) to solve the challenge, and by estimating the percentage of competitors they expect will be able to solve it. Challenges with a high experience level and the lowest expected solution rate are the most difficult.

Estimating the time it will take to solve the challenge can also help determine its difficulty. Challenges that are expected to take competitors much longer to solve can be considered more difficult due to the amount of effort required. Solution time estimates from testers are not always accurate when compared to true solution times, which vary depending on the competitor's level of expertise in a given subject. When testers estimate the solution times, they assume the competitor has an average proficiency in the knowledge and skills required to solve the challenge. However, competitors will solve challenges in their area of expertise more quickly than competitors with less experience. Although they are not always accurate, solution time estimates are useful as part of the challenge review process.

After testers complete their review of the challenges, competition organizers can examine the difficulty statistics and compare each challenge with others. Comparing the difficulty statistics with all challenges will ensure that easier challenges remain in earlier rounds and are worth less points than challenges that appear to be more difficult. When deciding challenge point allocations, organizers can use a base or standard score allotment as a starting point (e.g., all challenges are worth 1000 points at the beginning of the process). Organizers can then increase or decrease point allocations based on the available difficulty data, keeping in mind that the main goal is for the amount of points they allocate to a challenge to directly correspond with the effort required for

solving it. Point allocations should consider both the difficulty and the time it takes to solve the challenge.

Challenge Review Example

This example provides the difficulty and solution time statistics for three challenges along with their allotted point values. The review process in this example begins with a base or standard point allotment of 1500 points.

As part of challenge review, developers asked five testers to estimate the number of minutes required to solve a challenge and to assign a difficulty rating to the challenge. Difficulty ratings included Beginner (value of 1), Intermediate (value of 2) and Advanced (value of 3). Challenges between two difficulty ratings were assigned half values (e.g., testers assigned a challenge between the Beginner and Intermediate level a value of 1.5).

The *Solution Time Estimate* and *Difficulty Rating* rows in Table 1 below show the average of five testers' reports.

Table 1: Solution Time Estimate, Difficulty, and Point Allotment for Three Example Challenges

	Challenge 1	**Challenge 2**	**Challenge 3**
Solution Time Estimate	90 min	75 min	120 min
Difficulty Rating	2.3	1.8	2.5
Allotted Points	1500	1200	1800

Challenge 2 has the lowest solution time estimate and difficulty rating. Challenge 3 has the highest solution time estimate and difficulty rating. Challenge 1 falls between Challenges 2 and 3.

Because Challenge 2 has a lower solution time estimate and a significantly lower difficulty rating than the other challenges, developers decreased the point allotment below the standard allocation. This lower point value reflects the lesser difficulty and the smaller solution time.

Challenge 3 has a difficulty rating that is marginally higher than that of Challenge 1. However, the solution time estimate for this challenge is significantly higher than that of Challenge 1. Because of the marginally higher difficulty and significantly higher solution time, developers increased the point allotment above the standard allocation. The higher point value reflects a larger reward for the higher difficulty and estimated solution time.

The statistics for Challenge 1 fall between those of Challenges 2 and 3. Therefore, the allotted points remain equal to the standard allotment in this case.

5 Open Source Applications

There are several open source applications that developers can use to develop challenges and to orchestrate cybersecurity competitions. This section discusses the following two SEI-developed applications for running cybersecurity competition: (1) the TopoMojo lab builder and player, which developers can use to create and run challenges, and (2) the Gameboard platform, which developers can use to run a competition.

5.1 TopoMojo

TopoMojo is an open source lab builder and player application that developers can use to develop cybersecurity challenges [SEI 2022c, 2022b]. TopoMojo provides virtual workspaces where challenge development can take place. The workspaces allow developers to add virtual machines, virtual networks, and any other resources that are required for developing or solving a single challenge. Each workspace contains the challenge documentation and instructions, challenge questions, and other files that support the challenge.

Each challenge has its own workspace where it is developed. Developers make changes to the VMs and other materials in the challenge workspace and then save the changes when the challenge is in its final state. When a competitor starts a challenge, a read-only copy of the workspace, called a game space, is deployed for the competitor to interact with. The game space allows the competitor to interact with all workspace materials in a volatile environment, which the system destroys upon completion of the challenge (i.e., when time expires or the challenge is fully solved).

5.2 Gameboard

Gameboard is an open source application that organizers can use for orchestrating cybersecurity competitions [SEI 2022a]. It enables organizers to create competitions that can either be team or individual based and that consist of either single or multiple rounds. Challenges are organized into rounds and competitors attempt to solve as many challenges as they can to maximize their score. Gameboard uses the TopoMojo API to deploy the competitors' game space for each challenge. Gameboard also serves as the authoritative location for competitors to submit answers or tokens. Furthermore, as part of handling answer and token submissions, Gameboard has logging, brute force protections, and other features to ensure the integrity of the competition.

Figure 10 below illustrates how the TopoMojo and Gameboard applications interact. Developers use TopoMojo workspaces to develop challenges. Competitors use Gameboard to deploy and interact with challenges. When a player deploys a challenge, Gameboard will interact with the TopoMojo API to request a new game space for the competitor. TopoMojo creates and returns the player's challenge game space.

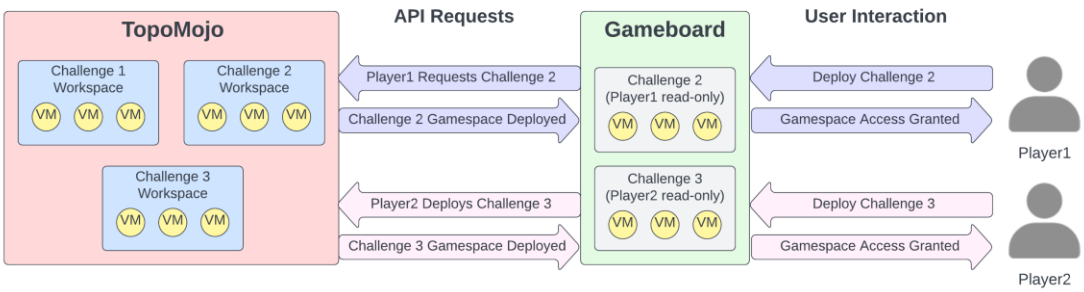


Figure 10: TopoMojo, Gameboard, and User interaction

6 Conclusion

Cybersecurity competitions provide a fun and interesting way to exercise technical skills, identify and recognized cybersecurity talent, and engage students and professionals in the field. Furthermore, cybersecurity competitions can serve as education and training opportunities. With the United States government and the nation as a whole facing a significant shortage in the cybersecurity workforce, cybersecurity competitions play an important role in expanding the workforce pipeline and developing talent.

There is no single way to run a competition, and there is no one way to develop cybersecurity challenges. However, there are some general best practices for developing cybersecurity challenges that this paper covers to help developers ensure the challenges they create are effective and engaging. These include ensuring that the challenges exercise the targeted knowledge and skill, and that developers have tailored them to the appropriate difficulty level. In addition, the best practices covered in this paper ensure that challenges do not contain shortcuts or dead ends; protect against answer sharing and cheating; award the appropriate number of points; and can be reused for future competitions or as learning resources.

This paper also covers best practices for developing the technical and supporting aspects of the challenge, including how to implement a grading mechanism that is automated and auditable; how to incorporate variability into challenge deployments; suggestions for writing supporting documentation for both challenge testers and competitors; and testing challenges thoroughly to make sure they will play out as expected. Finally, the paper covers some open source tools that developers and challenge organizers can use to develop and orchestrate cybersecurity competitions.

The challenge development practices outlined in this paper are the result of the SEI's experience developing cybersecurity challenges for the President's Cup Cybersecurity Competition. These practices provide a general-purpose guideline for developing effective cybersecurity challenges.

Challenge development is the single most important and time-consuming aspect of running a cybersecurity competition. Creating successful cybersecurity challenges begins with meticulous planning, continues with the technical development itself, and ends with a rigorous quality-assurance process. The results of this process are cybersecurity challenges that ensure successful competition execution and enduring, hands-on cybersecurity assets that competition organizers and others can reuse many times over.

Appendix A - Example Challenge Guide

The challenge guide example provided in this section is from a challenge that was part of the 2021 President's Cup Cybersecurity Competition.

Old is Gold

Analyze a forensic image to investigate an incident.

NICE Work Role:

- Cyber Defense Forensics Analyst

NICE Tasks:

- T0027—Conduct analysis of log files, evidence, and other information to determine best methods for identifying the perpetrator(s) of a network intrusion.
- T0396—Process image with appropriate tools depending on analyst's goals.
- T0532—Review forensic images and other data sources (e.g., volatile data) for recovery of potentially relevant information.

Background

Network operators at your organization have seen some ICMP data exfiltration from a particular system in the network. This system has also made a connection to a suspicious IP (1.66.20.98). Based on the intelligence data, a well-known text editor has been compromised and is responsible for connection to that suspicious IP.

An incident responder has imaged the system and the forensic image is available for analysis. According to the incident responder, the text editor in question was running on the system when the responder imaged it.

Getting Started

You are provided with two analyst workstations—Win10 and SIFT. The evidence ISO containing the hard drive image (image.dd) is attached to both the VMs. The incident responder forgot to acquire memory of the system.

Your end goal is to analyze the forensic image and answer the questions.

System and Tool Credentials

System	Username	Password
analyst-sift	user	tartans
analyst-win10	user	tartans

Submission

Submit answers to the questions below:

1. Provide the process ID of the process responsible for connection to the suspicious IP (1.66.20.98).
2. Provide the IP address that received the exfiltrated data.
3. Provide the MD5 of the file that was exfiltrated.
4. Name the executable that the attacker used to cover his tracks (delete other files).
5. What time did the attacker upload the file referenced in the previous question on this system?
6. What was the previous name of this file (the file referenced in the previous two questions)? You want the filename when it was uploaded to the system by the attacker.

Appendix B – Example Solution Guide

The following solution guide formed part of a challenge in the 2021 President's Cup Cybersecurity Competition. The contents from the example solution guide included in this section are abbreviated to show only a walkthrough for answering the first of six questions.

Old is Gold

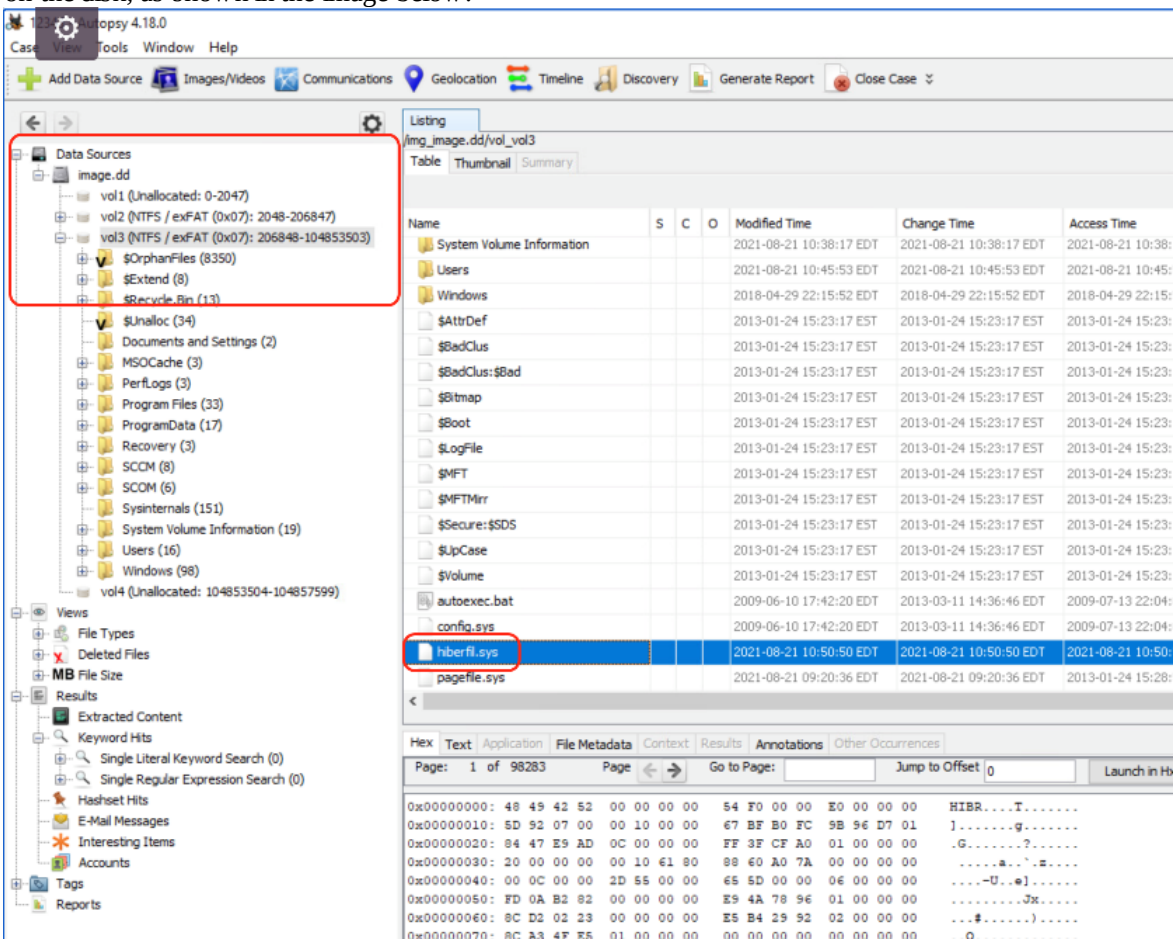
We are provided with a hard drive image of the system. Let's first mount it in Autopsy.

1. On the Windows analyst VM, open Autopsy.
2. Click *New Case*.
3. Give it a *Case Name*.
4. Change *Base Directory* to *Documents* folder.
5. Click *Next*.
6. Give it a *Case Number*.
7. Click *Finish*.
8. Select the type of data source to add as *Disk Image or VM File*, and then click *Next*.
9. Either browse to the evidence image or enter the path as *D:\image.dd*, and then click *Next*.
10. Click *Deselect all* to avoid running any ingest modules, and then click *Next*.
11. When the image is added to Autopsy (this process could take a few minutes), click *Finish*.

Q1 Provide the process ID of the process that was used to connect to the suspicious IP (1.66.20.98)

This kind of question is usually answered by analyzing a memory image, but as mentioned in the challenge description, the incident responder forgot to image memory of the system. Let's see if there is a *C:\hiberfil.sys* file present in the hard drive image since it is the compressed version of RAM.

1. Expand *Data Sources -> image.dd -> vol3*. You'll notice that the *hiberfil.sys* file is present on the disk, as shown in the image below.



2. Export this file. To do so, right-click on the file, click *Extract File(s)*, select *Desktop* as the folder to export it to, and then click *Save*.

The next step is to convert the *hiberfil.sys* file to raw memory. There are two tools present on the windows analyst system that can help with this conversion—*Hibernation Recon* and *Volatility*. *Volatility* takes a long time for this conversion. Therefore, let's use *Hibernation Recon*.

1. Open the *Hibernation Recon* folder on the desktop, and open *HibernationRecon.exe*.
2. Click *OK* to free mode enabled.
3. Click *Process hiberfil.sys*.
4. Select the previously exported *hiberfil.sys* from the desktop, and then click *Open*.
5. Select *Desktop* as the output folder, and then click *OK*.

Hibernation Recon is now decompressing *hiberfil.sys*. It will take a few minutes (~4 min) for the process to complete.

6. Once the conversion is complete, you can take note of OS, version, and architecture type information as this will be needed when processing raw image using *Volatility*.

The raw image is saved to a folder on the desktop named *HibRec-<date and time>*.

The next step is to analyze this image using *Volatility*.

1. Open the command prompt.
2. Change directory to *Volatility* location.

```
cd
C:\ProgramData\chocolatey\lib\volatility\tools\volatility_2.6_win64_standalo
ne\
```

3. Run *Volatility* and list the processes running in memory using the *pslist* plugin.

```
volatility_2.6_win64_standalone.exe -f C:\Users\User\Desktop\HibRec_2021-08-
23-09-48-56-21849\ActiveMemory.bin --profile=Win7SP0x86 pslist
```

You'll notice that only one text editor process (*sublime_text.exe*) is running. The process ID of this process is the answer to this question.

References/Bibliography

URLs are valid as of the publication date of this document.

[DHS 2021]
Cybersecurity Competitions. *Department of Homeland Security*. June 2, 2021. https://www.dhs.gov/science-and-technology/cybersecurity-competitions

[NICCS 2021]
Workforce Framework for Cybersecurity (NICE Framework). *National Initiative for Cybersecurity Careers and Studies*. July 29, 2021. https://niccs.cisa.gov/workforce-development/cyber-security-workforce-framework

[NICCS 2022]
NICE Cybersecurity Workforce Framework Work Roles: Cyber Defense Analyst. *National Initiative for Cybersecurity Careers and Studies*. January 6, 2022 [accessed]. https://niccs.cisa.gov/workforce-development/cyber-security-workforce-framework/work-roles?name=Cyber+Defense+Analyst&id=All

[Petersen 2020]
Peterson, Rodney; Santos, Danielle; Smith, Matthew C.; Wetzel, Karen A.; & Witte, Greg. *Workforce Framework for Cybersecurity (NICE Framework)*. NIST Special Publication 800-181, Revision 1. National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-181r1

[POTUS 2019]
Executive Order 13870: America's Cybersecurity Workforce. Executive Office of the President of the United States. May 2, 2019. https://www.federalregister.gov/documents/2019/05/09/2019-09750/americas-cybersecurity-workforce

[SEI 2022a]
Gameboard. *SEI GitHub*. January 20, 2022 [accessed]. https://github.com/cmu-sei/Gameboard

[SEI 2022b]
TopoMojo Software and Supporting Materials. Software Engineering Institute, Carnegie Mellon University. January 6, 2022 [accessed]. https://resources.sei.cmu.edu/library/asset-view.cfm?assetID=641154

[SEI 2022c]
TopoMojo. *SEI GitHub*. August 24, 2022 [accessed]. https://github.com/cmu-sei/TopoMojo

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	Month and Year (date added at time of publication)	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Challenge Development Guidelines for Cybersecurity Competitions	FA8702-15-D-0002

6. AUTHOR(S)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2022-TR-005

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
SEI Administrative Agent AFLCMC/AZS 5 Eglin Street Hanscom AFB, MA 01731-2100	n/a

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

14. SUBJECT TERMS	15. NUMBER OF PAGES
	34

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. Z39-18
298-102