The Center for Information Systems
Security Studies and Research

CyberCIEGE
Scenario Development Tool
User's Guide

June 2013

h t t p : / / c i s r . n p s . n a v y . e d u

TABLE OF CONTENTS

Introduction and Prerequisites

This guide addresses two general topics: the basic functions and features of the CyberCIEGE game engine from the perspective of a scenario designer, and the mechanics of constructing a scenario using the Scenario Development Tool (SDT). Designers construct scenarios using the SDT and compile them into a "scenario definition language". This language is consumed by the CyberCIEGE game engine. The game engine performs vulnerability assessment, network topology parsing and game economy management (e.g., assessing whether virtual users are achieving their goals).

The SDT automates the syntax of the CyberCIEGE scenario definition language through the use of reusable libraries and forms having pull down menus. The basic elements of this language can be reviewed via the SDT forms section of this manual.

Experience with the CyberCIEGE game is required. Play some sample scenarios and browse the CyberCIEGE Encyclopedia to learn about the game. If game behavior is directly relevant to the player, it will be described in the encyclopedia and not here. After you have experience with the game, read the section titled "Constructing a Scenario".

When you are ready to learn the mechanics of scenario construction, it is suggested that you read through the following sections to become familiar with the basic structure of the tool. Then follow the tutorial within this guide to learn some of the mechanics of using the SDT.

After you are familiar with the mechanics of constructing a scenario, read the section titled "Advanced Scenario Design".

Constructing a Scenario

Story telling is the key to a good CyberCIEGE scenario. The player should readily grasp the nature of the virtual environment (e.g., a small business with valuable intellectual property) and the types of choices that the player is to make. And the player should have reason to care about the ramifications of these choices.

Scenario designers utilize the scenario definition language to construct a virtual environment that drives players to make resource management decisions. These player choices affect the productivity of an enterprise, and affect the vulnerability of information assets to compromise by virtual attackers. The CyberCIEGE game engine interprets the scenario definition language, presenting the player with a virtual environment as defined by the designer. To construct a scenario, the designer must understand the semantics of the scenario definition language and the capabilities of the CyberCIEGE game engine. The SDT forms based integrated development environment allows designers to construct scenarios without mastering the syntax of the design language.

Programming the CyberCIEGE Game Engine

In every CyberCIEGE scenario, the player is the information assurance decision maker for some enterprise. An enterprise may range from a large military facility, to a home office. The fundamental elements within the CyberCIEGE game engine are not computers, networks and protection mechanisms – those are secondary. The primary elements are assets, users and attackers. Assets are information resources. Users are typically employees of the enterprise who have goals that require computerized access to assets. Players succeed by facilitating user access to assets. Some assets have substantial value to the enterprise based on secrecy or integrity. And some assets may have value based on their availability. Assets also have value to attackers, and this *motive* determines the *means* by which the attacker will attempt to compromise an asset. Thus, the motive determines the threat. Player choices affect the *opportunity* (or lack thereof) for the attacker to compromise the assets. Thus, player choices determine the system vulnerabilities. The enterprise (and by extension the player) is penalized the value of an asset should it be compromised or made unavailable.

Within any given scenario the users, assets and attackers are for the most part fixed by the designer and are not modified by player choices. Designers also specify the initial state of the scenario (e.g., an initial set of computers) and dynamic changes to the scenario (e.g., the introduction of new user goals.)

Players see the enterprise as an animated three dimensional representation of an office building or military headquarters. Each scenario has one main office and one or more optional small offsite offices. Users inhabit these buildings, wandering about or productively sitting at desks in front of computers. If computers are available, or purchased by the player, users will create and access assets using the computers. This user behavior is driven by the user goals specified by the designer. If computers are networked together, users may access assets over the network. Network devices such as routers enable users to access the Internet, and allow attackers on the Internet to potentially access enterprise resources. Suitably motivated attackers can enter buildings to compromise assets. They may compromise computer-based protection mechanisms, and may wire tap network links. Attackers may also bribe untrustworthy users to compromise assets. And users themselves may have motive to compromise assets.

Players may hire guards to help physically protect buildings or offices within buildings. And players may purchase physical protection mechanisms such as alarms and select which users are permitted to access different physical areas (i.e., "zones") within the virtual buildings. Procedural security choices affect user behavior (e.g., leaving computers logged in). Players can purchase user training to improve user adherence to procedural policies.

Illustrating the Engine with a Simple Scenario

Consider a scenario consisting of a single asset and a single user having a goal to read the asset. If this scenario is fed to the CyberCIEGE engine, the user will fail to achieve the goal of reading the asset until the player buys the user a computer. The designer associates a productivity value with the user that affects the size of the penalty resulting from failure to access the asset. When the player purchases a computer, the user will create the asset on the computer. Once the asset exists on a computer, attackers potentially target the asset. For example, an attacker might break into the office housing the computer and walk off with the entire computer. Or, if the asset's attacker motive is based on integrity, the attacker might hack into the computer and modify the data. If the asset is compromised, the player is penalized as specified when the designer defines the asset.

In the above example, the designer simply defines a user and an asset. The game engine manages the rest. The game engine manages the virtual economy to reward players when users are productive and to penalize when goals are not achieved or assets are compromised. And it includes sophisticated attack logic to assess the suitability of the current protection mechanisms to protect the assets based on the asset motive.

Extending the Scenario

In addition to defining assets and users, the designer specifies the initial state of the scenario including:

- Physical security properties (e.g., alarms, guards, etc.) of the different zones (e.g., offices);
- The set of pre-existing computers and their configurations including network connections;
- Procedural security policies to be followed by the users;
- User training (which affects adherence to procedural policies).
- Background checks for different kinds of users (e.g., based on user clearance);
- The timing and sequencing of attack types (but not the success or failure of attacks, that is up to the game engine, the initial scenario state and player choices.)
- How much money the player will start with;
- The kinds of computers and network devices available for purchase;
- Support staff available to help administer and maintain computer systems.

Attacks and the Tension Between Fidelity and Abstraction

CyberCIEGE includes a sophisticated attack engine that simulates a broad range of attacks, including Trojan horses, viruses, subversion of protection mechanisms, unpatched software flaws, insider attacks and physical attacks. On the one hand, the fidelity of CyberCIEGE is high enough to accurately represent these attack types and show the role of various protection mechanisms and policies in countering the attacks. (For example, the game can simulate an unpatched flaw in a particular service that is not being blocked by a firewall.) And on the other hand, the game components and configuration settings are abstract enough to let players make consequential choices without being overwhelmed with details of specific real-world products.

A primary goal of CyberCIEGE is to help students understand some fundamental cyber security concepts. To achieve this goal, the game does not need to individually represent every different application software flaw (e.g., each instance of a buffer overflow) that leads to execution of arbitrary code. Similarly, the game does not need to individually represent every application's failure to perform input checks (e.g., SQL injections) and it does not need to simulate every different Trojan horse or virus.

By maintaining a consistent level of abstraction, the game is able to represent a rich collection of attacks and countermeasures. Instead of learning that a particular version of a specific web server has a particular buffer overflow flaw, the player learns about the need for patch management, configuration management, access control mechanisms, and the risks of relying on applications of uncertain pedigree to protect valuable assets. The game engine lets scenario designers define different applications in terms of the software's integrity (e.g., what role did software security play in its development and distribution?), and in terms of its need for patches. The game engine also lets developers express the goals of users internal to the simulation in terms of a need to use a particular type of software, and whether the access is remote (i.e., is subject to network filter configurations). This allows scenario designers to create in-game tensions that steer the player towards an understanding of the fundamental choices and risks associated with cyber security.

User Animations and Interactions

The game characters have a limited number of animated actions. In general, if a user has an assigned workstation, the user will sit at it and work. If the user is failing to achieve goals, the user will pound the keyboard. If the user lacks an assigned computer, it may wander the office, or if some unassigned computer can be used to achieve goals, the user will sit at that computer. Depending on the user (wander) and scenario (ignore daylight) settings, users may also wander periodically while they are achieving goals. Users don't necessarily interact with all of the computers that they use while achieving goals – if they have assigned computers, they will only interact with those. If not, they will pick one computer that lets them achieve goals and interact with that. Users create assets as needed, but again they are not necessarily interacting with associated computers when creating assets. In other words, user interactions with assets are not synchronous with their animations.

Users interact with zone checkpoints. When checkpoints are defined (see the Zone Form), a user cannot enter the zone without passing the checkpoint, and this may require interacting with a device (e.g., card reader, iris scanner, etc.). User entry to zones is synchronous with the animations. Note however that a user does not have to be physically inside the zone to be achieving asset goals – the user only needs to be able to enter the zone.

Triggers can be used to cause virtual users to "visit" other users, or go to the location of other user's computers. And triggers can adjust a user's happiness, and thus the icon the appears above the user when the player clicks on the user.

Interacting with the Player & Dynamically Altering the Scenario

The CyberCIEGE scenario definition language allows scenario designers to periodically assess the ongoing game state "conditions" and respond using active "triggers". Game state conditions include such things as the passing of time, whether users are achieving their goals, computer configuration settings and whether attackers have compromised assets. Active triggers include popup messages, brief movies, changes in user goals, commencement of attacks, and user feedback to the player via balloon speech as reflected in figure 2.

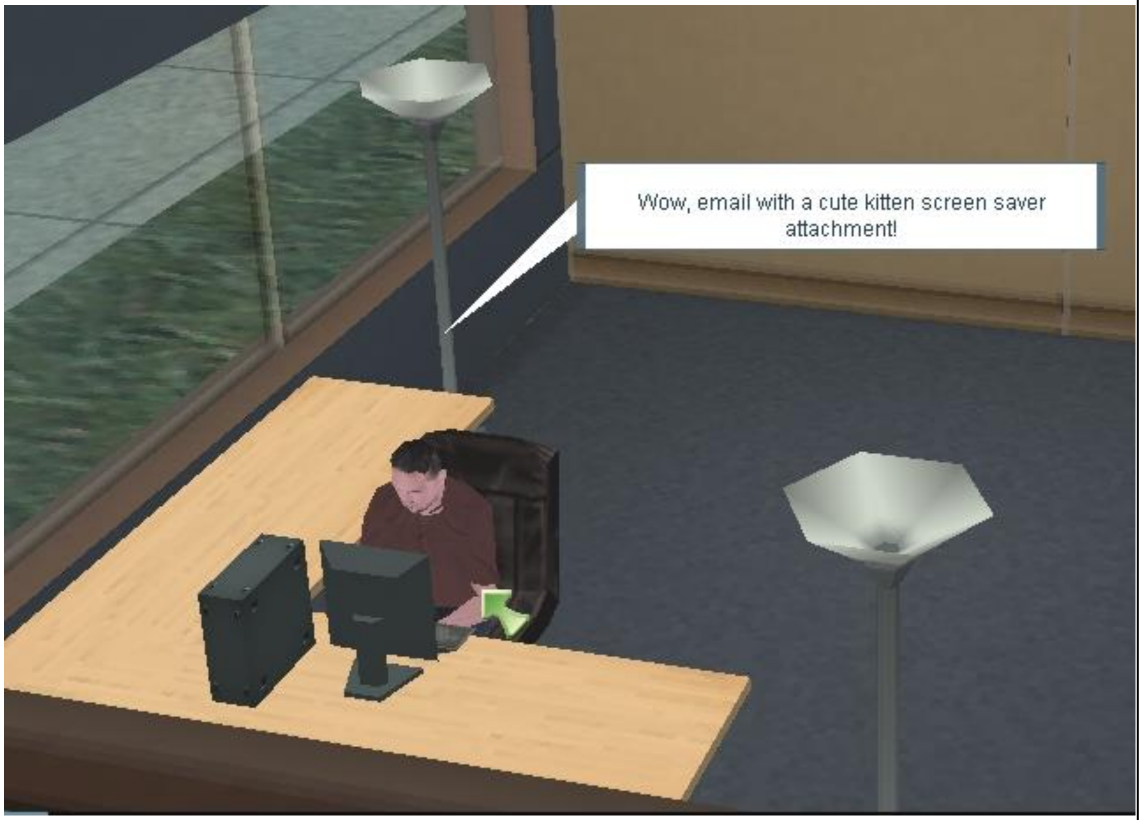


Figure 1: Balloon Text

Scenarios are divided into multiple phases, each of which includes one or more objectives that the player must achieve prior to moving on to the next phase. Designers use conditions to assess whether objectives have been met and designers use triggers change the environment for the next phase (e.g., introduce additional user goals).

The SDT includes a rich set of conditions, however not all condition types need to be used within a scenario. Many of the conditions are intended to allow the designer to provide the player with feedback *before* the engine does harm to the player's network. For example, the condition that measures the number of open application ports in a router can be used to warn the player. Use of that (or any other) condition is not necessary to cause the game engine to attack assets. Similarly, some conditions measure a user's ability to achieve a goal. Use of this condition is not needed to penalize the player for unproductive users – rather it can be used to provide the player with specific help.

Selecting the Scenario Audience

The first step in the design of a scenario is to identify its purpose and audience. For example, does the intended audience have experience with computer games? Are they expected to have played other CyberCIEGE scenarios and thus have some level of mastery of the mechanics of the game?

The scenario definition language supports a broad range of different types of scenarios. At one end of the spectrum are simple scripted scenarios such as those intended for basic training and awareness. These training and awareness scenarios are designed to affect user behavior where human factors are the source for potential security compromises, e.g., "beware of email attachments." This type of scenario is often built entirely from conditions and triggers, with very little reliance on the game engine's economy or attack engines. For example, a set of conditions assess whether the user has been instructed to beware of email attachments, and triggers provide direct feedback based on that game state. At the other end are sophisticated scenarios for players who have a basic understanding of network security engineering. These scenarios rely more on the game engine itself to direct attacks and manage the overall economy.

Scenarios can be viewed as belonging to one of three classes:

1) Training and awareness – generally ignore the economy engine and attack engine. Almost entirely driven by triggers and conditions. May incorporate game features as a way to introduce players to more sophisticated scenario types (which is the primary reason such scenarios might be build in CyberCIEGE rather than Flash.)
2) Educational – Varying levels of player feedback and help, but generally intended to illustrate one or more major security concepts (e.g., use of VPNs.) Scenarios designers need to exercise care to keep players from being overwhelmed. Attacks are carefully sequenced. Some random feedback and messages, but generally these scenarios will play out the same way each time the play makes the same choices.
3) Free play – Player feedback generally limited to moving the story along and creating depth. Attacks can occur randomly and without regard for overwhelming the player. Complexity achieved by adding new asset goals and users as time progresses. These scenarios have the potential for substantial random events and thus will tend to play differently each time.

Elements of Scenario Design

The scenario designer defines the information assets. What kind of information is it? What is the asset value and what makes it valuable? Why would an attacker target the asset? Asset values and attacker motives are set in the Asset form. They can also be set indirectly via the Secrecy and Integrity forms for assets that have security labels.

The designer also defines the users. What assets do the users need to access? Why do they need to access them? Do users need to share assets? Do users require access to assets via the Internet (e.g., publicly available documents)? Do users have to use specific kinds of software when accessing certain assets (e.g., web browsers, word processers, etc.) User goals are defined in the Goal forms. These goals are then assigned to individual users in the User forms.

The scenario designer describes the story line in the scenario briefing and in the descriptions of the assets and users. The Phase and Objectives forms are used to divide a scenario into multiple phases, with each phase having one or more objectives that must be met to advance to the next phase. The designer describes individual player objectives and specifies the conditions that constitute the achievement of each objective. Players typically view information about their current objectives by viewing this text in the Objectives screen. All textual information is intended to provide players with the context of the scenario.

The initial state of the scenario can be used to constrain a player's options. For example, a player can be given insufficient cash to fully secure a site until an initial set of objectives are achieved.

Attack timing and sequencing is determined by the scenario designer. Triggers generate the different types of attacks based on conditions set forth by the designer. Note again that the designer generally does not decide if an attack should succeed, that is the based on asset motive and player choices to address vulnerabilities. On the other hand, the designer may want to evaluate the player's network to determine if an attack would succeed so that the player can be warned. Several different kinds of conditions are included to achieve that (e.g., AssetToNetwork).

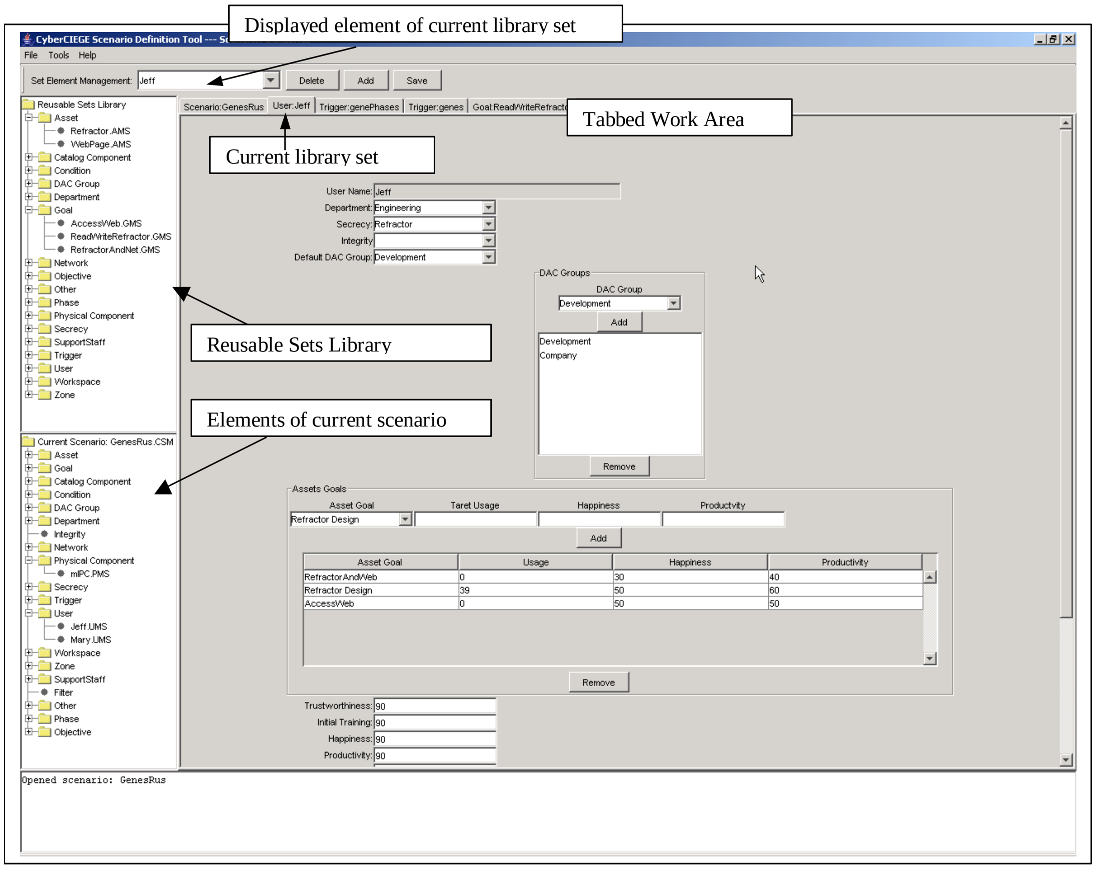
The designer defines feedback to move the player along through the scenario based on current game conditions. For example, what should a virtual user say or think if a specific goal cannot be met? The engine causes the user to wander aimlessly or violently pound on the keyboard. The designer can enhance this with specific user "thoughts" or comments that appear in bubble text. In some scenarios the designer may choose to assess suitability of protection mechanisms using conditions and warn the player prior to the attack engine's exploitation of the vulnerability. And in other scenarios the designer will provide substantial help tips to aid the player with the mechanics of the tool. CyberCIEGE includes a rich on-line encyclopedia that can serve as context-dependent help through the use of triggers. Ultimately the designer selects the conditions that constitute a "win" or a "loss", and provides the text to display in the respective debriefing. The encyclopedia includes several animated movies tutorials (e.g., describing malicious software) that can be launched as part of the debriefing.

As you develop scenarios and try them out on the game engine, keep in mind that the game engine is not intended to provide a game-play experience to the designer via the SDT. Rather, it is the combination of the scenario design and the game engine that creates the game-play experience via the game interface. In other words, you can create scenarios that result in inconsistent game play.

See the section on "Advanced Scenario Development" for additional guidance on creating scenarios.

Using the SDT

The following sections describe the layout of the SDT and its use in constructing scenarios.



Set Element Management: Jeff

User Name: Jeff
Department: Engineering
Secrecy: Refractor
Integrity
Default DAC Group: Development

DAC Group: Development

Development
Company

Asset Goal	Taret Usage	Happiness	Productvity
Refractor Design			

Asset Goal	Usage	Happiness	Productivity
RefractorAndWeb	0	30	40
Refractor Design	39	50	60
AccessWeb	0	50	50

Trustworthiness: 90
Initial Training: 90
Happiness: 90
Productivity: 90

```
Opened scenario: GenesRus
```

Figure 2: Scenario Development Tool Screen Layout

Reusable Sets Library

The upper left window contains folders for each set type (e.g., "User"). Each folder can contain a multiple sets and each set can contain any number of elements. Left clicking on a folder (e.g., "Asset") and selecting "new" creates a new library set. Double clicking on a set within a folder will open the set in the tabbed work area. The "File/Save As" menu item creates copies of entire sets.

Reusable Set Elements

When a reusable set is opened, it is added to the tabbed work area. When a set's tab is selected, one element of the set is displayed. Use the "Set Element Management" pull-down list to select the element to display and/or edit. Click the "Add" button to create a new element having initial values copied from the previously displayed set element. New elements are created beneath the previous element, which is useful when defining order-dependent elements such as triggers and phases.

Scenario

The lower left window contains the sets contained in the current scenario. Only one scenario is "open" at a time. The left-most tab in the tabbed work area is always the scenario form. The current scenario can be changed via the "File/Open Scenario" menu. Add a library set to the current scenario by right clicking on the set in the Reusable Sets Library, or by right clicking on the tab if the set is displayed in the tabbed work area. All elements of the selected set are added to the scenario.

Remove element sets from the scenario by locating them in the scenario windowpane and right clicking on them and selecting "remove".

Menu Bar

The menu bar includes the following functions:

File

> Open selected scenario; save scenarios under different names; save all open scenario element sets, etc.

Tools

> Build
>
> > Compiles the current scenario into a Scenario Definition File for consumption by the CyberCIEGE game engine. First the SDT performs basic validation of the current scenario looking for elements that are referenced but not defined. Errors are displayed within the feedback text area at the bottom of the screen. Double clicking on a line will open that scenario element within the tabbed work area.
>
> Run
>
> > Start the CyberCIEGE game engine using the most recently built scenario in this project.
>
> Project Settings
>
> > Select current project; set current game directory; select editor; define run-time configuration switches. Runtime switches include:
> > -A Enable generation of camera viewpoints
> > -J Used when making screen-capture movies of game play to hide the hourglass cursor.
>
> Clone Project
>
> > Create a new project directory that is identical to the current project, but with a new project name.
>
> Build / Run
>
> > Build and run the current scenario.

View

View SDF

Display the raw scenario definition file created from the latest build.

View Log

Use the log viewer to display the game engine log generated from playing this scenario. See, "Debugging, Game Logs and Replaying Scenarios".

View Parse Errors

The scenario validation performed when building a scenario is not complete. Some errors are not caught until the game engine starts (i.e., using "Tools / Run". These kinds of errors are often displayed in the initial briefing page, and also are captured in this file.

View Crash Text

When the game engine abnormally ends, messages are written to the crash.txt file within the game/exec directory. Send this file to cyberciege@nps.edu.

Testing – under development. For now you can clear the log and run a set of test scenarios (e.g., select the "nt" project). At the prompt, enter the starting letters of the tests to run, or leave blank to run all tests. Build all must be run many times to make it through rebuild of all the tests because java garbage collection is garbage.

Search

Searches the libraries of the current project for a given string. Results are displayed in the feedback text area at the bottom of the screen. Double clicking on a line will open that scenario element within the tabbed work area.

Projects

The Reusable Sets Library window pane displays all reusable sets within the current project, regardless of whether they are in the current scenario. A different project can be selected using the Tools/Project Settings menu.

The preferred way to create a new project is to copy an existing project using the "Tools/Clone Project" menu. First select the project you wish to copy, and then select "Tools/Clone Project". In the file dialog, create and select a new directory having the name you wish to use for the project.

1) If you select (or create) an empty folder, the tool will create all the necessary subdirectories. To populate a scenario, use the SDT to create new sets of elements and add them to the scenario;

Please note that re-usable sets libraries are not implicitly shared between different projects. They are intended to be shared between scenarios within the same project.

Building Game Scenarios

Build and test your scenarios incrementally to limit the scope of possible errors. CyberCIEGE error checking and reporting is not as robust as it should be, and the SDT lacks an "undo / redo" function, so be prepared to back out changes to get back to a working version of your scenario.

It is strongly recommended that you follow the tutorial section of this document when creating your first scenario.

Running Game Scenarios

Before running a game, use the "Tools/Build" function to check for references to non-existent set elements (e.g., assigning a non-existent user to a component). Use "Tools/Run" to execute the result. Running the game requires that the "Tools/Project Settings" Game Directory point to the games "exec" directory. The SDT will copy the SDF into a file called "tmp.sdf" and it will copy the newly created workspace file into the game directory as well.

"View/SDF" displays the SDF built via the "Tools/Build" function using the viewing program named in the "Tools/Project Settings" Viewing program entry. This is initially set to "WordPad".

Note that error checking is limited. Use the Tools menu to view the game log and the "ParseErrors.txt" file created in the game directory when the game detects errors in the SDF. Finally, view the "crash.txt" file from the "View/crash.txt" menu. If the final entry in that file reflects the current date and time, it may give you a hint as to what kind of error the game encountered.

Debugging, Game Logs and Replaying Scenarios

The game log viewer lets you see each game event that occurred during the course of playing a scenario. The log viewer lacks a full user's guide, however most of its functions are straight forward. Be sure to review all the "Filter" settings when looking for any given log entry to make sure you are displaying it. Change the filters and click "Apply" to view a revised subset of log entries.

Some (but not all) "AssetAttacked" log entries include "reason" properties. Double click on these to see the entire text.

Log entries for trigger events include the condition values that set off the trigger – review those Boolean values when debugging trigger firing.

After a scenario has been played (from the SDT), the resulting log can be "replayed". This is primarily intended for use when debugging and testing scenarios. Using replay lets the developer avoid manually repeating click/key sequences necessary to get to a desired point within a scenario. Note that only player choice events are replayed from the log. The game engine responds to the replayed choice events just as it responds to any player choice. Note that randomness in the game engine and scenario definitions prevent the resulting game from being identical to the original play – but it is usually close enough to serve the intended purpose. To replay a log:

> > Select the log to replay by browsing for the log file in the "Replay Log File" field (in Tools / Project Settings).

> To initiate replay, simply run the scenario again. After the initial brief screen, press the ">" key. The resulting window looks like the log viewer, and it has an initial filter that displays only user choice events. The following commands work while the replay window has focus:
>
> > Double click an entry sets / resets a breakpoint
> > F5 run to the next breakpoint
> > F10 single step (note not all events might be displayed, so be prepared to issued multiple F10's to get to the next visible log entry.)

> You can make choices within the game itself during a replay. The resulting log can then be the basis for a future replay. Note you must take care to ensure the proper window has focus prior to issuing key commands (e.g., the "c" compress time command requires that the game window have focus.)

As you build scenarios, consider building triggers that serve as test jigs, i.e., that only run while testing. For example, a test jig might move the scenario to the next phase even though the typical objectives have not been met. This lets you test later phases without playing through the early phases (either manually or via the replay feature). To simplify use of test jigs, take care to constrain other game triggers to firing in their appropriate phase (e.g., even though a given trigger designed for phase 1 would not normally fire in phase 3 during a normal game play, the use of a test jig might jump the game to phase 3 causing the trigger to fire and disrupt testing. In that case, an otherwise superfluous condition within the trigger can prevent the trigger from firing.) Also, avoid using slave triggers within the actual SetPhase triggers so that your test jig phase transitions mirror the game transitions.

Tutorial

This tutorial walks you through the creation of a CyberCIEGE scenario using the Scenario Development Tool. This tutorial does not define the semantics of the selected values. Refer to the form descriptions in this manual for a definition of the various parameters.

1) Establish a Project
Use Tools / Project Settings to select the "StartOffice" scenario. This is a very simple scenario with two users. Then use Tools / Clone Project to create a copy of this scenario. In the file dialog, create and select a new directory called "MyStartOffice".

The "MyStartOffice" directory is now where your new project resides. The above step is important because changes you make in this directory will not be over-written by CyberCIEGE updates.

3) Review the project's scenario elements.
Expand the folders within the scenario elements tree (lower left) to see the initial state of the example scenario. You will see an extremely simple scenario populated with several elements to help get you going without too much effort. The scenario includes:

- two users who have no goals
- some networks (just the wires)
- catalog components (components the player can buy)
- a simple time condition
- a condition that measures whether an attack occurred
- a few attack triggers that will fire after an hour of game time. The trigger conditions ensure that the attack types don't overlap, i.e., the second attack type won't occur until the first attack type fails to result in an asset compromise.
- zones
- procedural settings to establish default values for zones and components

2) Run the new project
Select Build/Run from the Tools menu. When the game starts, click the "Play" button. Note you initially have two users. You can see these two users defined in the User folder (either the Reusable Sets Library or the Current Scenario sets.) The users currently have no goals, so they will just wander and fidget.

2) Add an Asset
Right click on the Asset folder in the Reusable Sets Library pane. Select "new" and give the set a name, e.g., "Joe's Assets". Then give the first asset a name, e.g., "Joe's Spreadsheet". This results in a display of the asset form. Fill it out as follows:

- Give the new asset a description.
- Add an "Intended Access Control List" entry that permits Joe to read and write the asset (select Joe from the user pull-down list and then click "ADD").

- Add a "Cost List" entry that reflects a cost of $1000 and a motive of 50 should "Public" be able to read this asset. (enter "1000" in the "Cost" field and "50" in the "Attacker Motive" field and click "ADD").
- Add the newly created Asset Set to the scenario (right click on the tab and select "Add Tab To Scenario"

3) Add an Asset Goal
Right click on the Goal folder in the Reusable Sets Library pane. Select "new" and give the set a name, e.g., "Joe's Asset Goals". Then give the first asset goal a name, e.g., "Modify Joe'sSpreadsheet". This results in a display of the asset goal form. Fill it out as follows:

- Give the new asset goal a description.
- Add the "Joe's Spreadsheet" asset to the "Asssets Defined for this Goal". (Select "Joe's Spreadsheet" and select "Y" for read and write. Then click "ADD"
- Add the newly created Goal set to the scenario (right click on the tab and select "Add Tab to Scenario".

4) Give Joe the new goal
Expand the User folder in the Reusable Sets Library pane (if it is not already expanded). Double-click on the set named "Joe". That should display the user form already populated with user Joe. Give Joe the goal by adding an entry to the "Asset Goals" table. The "Modify Joe's Spreadsheet" should be the selected goal. Enter a value of 50 for each of the "Asset Usage", "Happiness" and "Productivity" fields and then click "ADD".

5) Run the Scenario
Select Build/Run from the Tools menu. When the game starts, click the "Play" button. What are Joe's thoughts? Double click on him. Is he achieving his goal? Why not? Go back to the OFFICE tab. Why is there a penalty?

- Use the buy button to purchase a workstation and place it on Joe's desk.
- Is Joe now achieving his goal? Notice how Joe has created the asset on his computer.
- Has the penalty become a bonus?
- Eventually the asset will get compromised. When that happens exit the game.

6) Review the log
Select "View Log" from the SDT "View" menu. Turn off the "Hide Attack Triggers" check box at the bottom and click "Apply". Find the AssetAttacked event toward the bottom of the log. What happened? Note the attack was preceded by an "ATTACK_TRIGGER" trigger. Your initial example scenario includes a few periodic attacks. We will refine those later in the tutorial.

7) Rerun the Scenario and Prevent the Break-in
Play it again and double-click on the zone containing Joe's workspace. Alter the Zone Access list so that only Joe is permitted to enter the zone. And buy a key lock. Notice the resulting "Phys Security" value. Since it now exceeds the motive of the asset, the physical attacks should now fail. Buy the workstation, unpause the game and speed

things up. No breakin right? Run the game until the user starts complaining that his machine keeps crashing.

8) Trigger player feedback

You should have noticed the computer was crashing, and maybe you noticed its availability went down. Why was that? Right click on the computer and run a malware scan. How did that get on there? Select "View" / "View Log File" from the menu bar. Uncheck the "Hide Attack Triggers" and click "Apply". Review the log. Notice that the "malware" warning event occurs after the "Bad Policies" attack trigger. Also note that the "Bad Policies" attack trigger will not fire unless the "Breakin-Hacking" attack trigger fails to result in a compromise. This logic is dictated by the conditions of the Attack Triggers initially defined in your scenario. Now lets give the player a bit more obvious feedback about the presence of malware.

- Right click on the Condition folder in the Reusable Sets Library pane. Select "new" and give the set a name, e.g., "Component Issues". Then give the first condition a name, e.g., "JoeMalware".
- Select the Condition Class of "VirusPresent" and Joe as the username. Add the condition to the scenario.
- Right click on the Trigger folder in the Reusable Sets Library pane. Select "new" and give the set a name, e.g., "Attack Response". Give the first trigger a name, e.g., "Burn Joe's Computer".
- Select a trigger class of "BurnComputerUser", set the delays to zero and set the user to Joe and set "Duration" to six.
- Define the firing condition as "JoeMalware". Add the trigger set to the scenario.
- Rerun the scenario, protect against break-ins and watch Joe's computer burn.

8) Prevent malware (at least in this scenario) – Procedural Settings & Zone Defaults

- Replay the game, preventing physical break-ins per the above. After buying the component, go to the COMPONENT screen and select "No External Software" from the procedural list. Speed up the game (<ctr> c). No malware right?
- Run the game again, but this time, before buying the component, go to the zone that will contain the component and set the "No External Software". This becomes the default for all machines introduced into that zone. Now buy the component. Look at its settings and note how "No External Software" is checked.
- In the SDT, look at the zone form for the zone into which you were placing the computer (expand "Zone" in the scenario elements list and double-click on "3a". Then select the zone, e.g., "lower left", from the "Set Element Management" pulldown list.) Find the procedural settings entry. It references a procedural settings set name (to the left of the %) and element of that set (to the right of the %".
- Find the procedural setting entry. Procedural settings are under the "Other" folder. Expand that and then expand the Procedural Settings folder. Then open the "default" set. Select the "default" element from the "Set Element Management" if it is not already selected. Check the "No External Software" entry. Now build and run the game. Look at the zone (e.g., lower left") and notice how the "No External Software" procedural settings is now set.

- You have just reviewed one of the many ways in which the scenario designer can control the initial conditions of the scenario.
- We'll leave the zone vulnerable to break-ins so that we can observe the attack logic later in the tutorial. You can undo the "no external software" change later to let players see your burning computer.

9) Add a Web Asset
This step is the first of several that will provide Joe with a goal that can only be achieved if he has an Internet connection.

Right click on the Asset folder in the Reusable Sets Library pane. Select "new" and give the set a name, e.g., "Web Assets". Then give the first asset a name, e.g., "Web Page". This results in a display of the asset form. Fill it out as follows:

- Give the new asset a description.
- Add an "Intended Access Control List" entry that permits Public to read the asset (select Public from the group pull-down list and then click "ADD").
- Add a "Cost List" entry that reflects a cost of $0 and a motive of 0 should "Public" be able to read this asset. (enter "0" in the "Cost" field and "0" in the "Attacker Motive" field and click "ADD").
- Add the newly created Asset Set to the scenario (right click on the tab and select "Add Tab To Scenario"

9) Add an Offsite Physical Component
Put a server in the offsite server rack:

- Expand the Workspace folder in the Scenario Elements pane and open the "WorkspaceStartOffice" set. See the "Workspaces" section of this manual. Then run the game and press the "g" key and view the offsite office (e.g., via the "l" [lower "L"] key). Notice that the server rack in the offsite office is at about grid position 95, 20. Now look at the Workspace table in the form. Scroll down until you find an X-Y entry that is close to 95, 20 and is a server rack (i.e., "S" in the Furn column). That entry says there is a server rack at that X, Y location, which we know is in the offsite office. So we will put the server there. Make note of the index value (e.g., "16"), for that is how we will specify the location of computers and users.
- Right click on the Physical Component folder in the Reusable Sets Library pane. Select "new" and give the set a name, e.g., "Offsite Components" and give the first element of the set a name, e.g., "Web Server". Fill in the form as follows:
    - Enter a description, e.g., "A remote web server."
    - Enter the position index noted above (e.g., "16") as the "Grid Position in Office"
    - Select the "Web Server" as the Procedural settings.
    - Find the "Base Component for this Device" and select "Blato Server" from the list". Select "Populos V9 Server" as its Operating System.
    - Scroll down to "Network Connections", select "Offsite LAN" and click the >> button.
    - Scroll down to the "Assets", select "Web Page" and click the >> button.

  - Right click on the tab to add the set to the scenario.
- We have made a lot of changes. Make sure things work as expected. In "Tools" select "Build / Run". When the game starts, press the "l" key to toggle between the main office and the WWW office. When the WWW office appears confirm there is a computer in the server rack. Select the Components tab and confirm the Web Server contains the Web Page asset. (Note the WWW office has no walls. Other offsite offices look more like small offices.)

10) Add a router to the offsite server rack.

- Select the "Offsite Components" tab and click the "Add" button in the Set Element Management bar. Give this second element of the set a name, e.g., "Offsite Router". (Again, note that sets may have more than one element. There is no semantic difference between sets having one and multiple elements. It is entirely up to the developers preferences for packaging the scenario.) Fill in the form as follows:
  - Enter a description, e.g., "A router."
  - Note the "Add" function causes the new element to inherit values from the previous element. So our Grid Position is already 16. But we'll have to change some of the values as described below.
  - Ignore the Procedural Settings for devices.
  - As the "Base Component for this Device", select "Bit Flipper" and then "Flip OS" as its Operating System.
  - Scroll down to "Network Connections", select "Internet" and click the >> button.
  - Remove the Web Page asset from the router
  - Build and Run. Select the NETWORK tab and look at the "WWW" zone. Confirm you've created a simple network.

10a) Create a new Asset Goal for Joe.

- Select the "Joe's Asset Goals" tab. Click "Add" and give the new set element a name, e.g., "Read Web Page". Give it a description.
- Select the "Joe's Spreadsheet" goal and click "Remove". (Note this extra step is a drawback of the set inheritance feature described earlier.).
- Select "Web Page" from the list of assets and click the ADD button.
- Click the "Save" button to make the new goal selectable from other forms, such as the next step. (As you build scenarios and you encounter a pull-down list that does not include one of your scenario elements, you may need to return to that element's form and click "Save".)
- Then give this goal to Joe by going to the User:Joe form and select values of "50" for target usage, happiness and productivity. Joe now has two distinct goals.

11) Build and run the game.

Unpause. Check Joe's goals. Not met right? Buy him a computer. One goal met. Now buy him a router:

- Click the "BUY" button and select the "Network Devices" tab.

- Select the Bit flipper router and click the BUY button. Place the router on Joe's desk, or perhaps in the on-site server rack.
- Click the "NETWORK" tab. Find the router. Select it and then click the "LAN1" and then the "Internet" buttons.
- Find Joe's computer. Select it and click the LAN1 button. You now have a simple network.
- Check Joe's goals. All met?

14) Add a condition to assess asset goals

Right click on the Conditions folder in the Reusable Sets Library pane. Select "new" and give the set a name, e.g., "Goals". Then give the first condition a name, e.g., "JoeFailsSpreadSheet". This results in a display of the condition form. Fill it out as follows:

- In the "Condition Class", scroll down and select "UserFailsGoal"
- Select "Joe" as the user, and "Modify Joe's Spreadsheet" as the goal
- Add the condition set to your scenario.

15) Use a ticker to remind the player to give Joe a computer

Right click on the Trigger folder in the Reusable Sets Library and select "new". Give the set the name "Messages" and give the first trigger element a name e.g., "Joe Computer Ticker" This results in a display of a new trigger form. Fill it out as follows:

- In the "Trigger Class", scroll down and select "TickerTrigger"
- Enter "0.05" for the frequency (the units is days)
- Enter zero for the fixed and random delay.
- Type a message to display to the player in the "Text" field, e.g., "Joe needs a computer…"
- Set "Repeat" to "0".
- Set "Erase" to 1. That will cause any tickers to disappear if the player buys Joe a computer so that the player is not confused into thinking Joe is still failing his goal.
- In the "firing condition", enter: "OneHour AND JoeFailsSpreadSheet"
- Add the ticker set to your scenario.
- Build and run. Use the "c" key to speed up time. Your message should pop up after about 1 hour of game time. The ticker should start then. And it will repeat about ever 0.05 days.
- Buy Joe a computer. The ticker should disappear as soon as he achieves the goal.

16) Make Joe speak (Comic book style anyway)

Select the "Trigger:Messages" tab and click "NEW" and give the new trigger a name, e.g., "Joe Computer Thanks" This results in a display of a new trigger form. Fill it out as follows:

- In the "Trigger Class", scroll up and select "SpeakTrigger"
- Enter 999 as the frequency (this means "once per 999 days", i.e., just once).
- Enter zero for the fixed and random delay.
- Enter "Joe" as the user field

- Enter "10" in the "oneForFreeze" field, this will display the speaks for about ten seconds.
- Enter the words Joe is to speak in the "Text" field, e.g., "Thanks for the computer!"
- In the "firing condition", enter: "NOT JoeFailsSpreadSheet"
- Build and run. Unpause. Buy Joe a computer. And you thought it was a thankless job.

17) Delay Joe's Internet Goal

User goals can be hidden from players until a set of conditions are reached. This lets you structure the scenario so the player does not need to do everything at once. Select the User: Joe tab and change his "Read Web Page" goal "Usage" to zero. (Be sure to hit the enter key to make sure the change is recorded.) Then add a trigger to give Joe a non-zero usage of this goal after he has access to his spreadsheet:

- Right click on the Triggers folder in the Reusable Sets Library pane. Select "new" and give the set a name, e.g., "Set Goals". Then give the first trigger a name, e.g., "Add Joe Internet". This results in a display of the trigger form.
- In the "Trigger Class" scroll down and select "ChangeAssetUsageTrigger".
- Enter zero for the fixed and random delay.
- Enter "Joe" in the user field
- Enter "Read Web Page" in the goal field (yes, these should eventually be pulldown lists)
- Enter 50 as the new target usage
- In the "firing condition", enter: "NOT JoeFailsSpreadSheet"
- Add the new trigger set to the scenario
- Build and run. Fix the physical security of his office. Unpause. Check Joe's goals. He only should have the one. Buy him a computer. Then check his goals. He should now have two goals, one of which (Read Web Page) is not being met.
- Buy a router and hook Joe up to the Internet. His goal should be met. Run until his computer bursts into flames. Then review the log to see what happened. Hooking him to the Internet enabled new vulnerabilities.

18) Review Attack Triggers

- This example scenario had three initial attack triggers. Look at them now. The first causes attackers to break into the office and compromise assets. The second causes malicious software to be installed on computers due to poor configuration choices or user procedures. The third causes Internet-based attacks.
- Note the frequency value limits each attack to occur no more than about once an hour.
- The Attack Type is per the values defined in the Attack Types section of this guide.
- The AttackMotive value of -3 tells the game engine to base the motive on the value associated with the asset itself (in this case, 50). The value of -3 also instructs the engine to not trigger an attack until a period subsequent to the most recent asset compromise as defined by the trigger's frequency. Thus, as long as

break-in attacks succeed, the scenario will not generate bad policy attacks or internet attacks. Note however that the bad policy attacks don't compromise assets, so their success does not limit the occurrence of Internet attacks.

- The section titled "Managing Attacks" provides additional information on the art of managing and responding to attacks.

20) Structure your scenario into phases. This is the first of several steps that will structure the existing scenario into two different game phases.

- Right click on the Phases folder in the Reusable Sets Library pane. Select "new" and give the new set a name, "MyPhases". Give the first element a name, "Phase One".
- In the form, assign values as follows:
  - Display Name: Work locally
  - Completed Text: Joe can work locally
  - Uncompleted Text: Provide Joe with a safe work environment.
- Click the "Add" button to add a second phase to this set. Name it "Phase Two" and fill in the form as follows:
  - Display Name: Access the Internet
  - Completed Text: [none]
  - Uncompleted Text: Give Joe access to the Internet.
- Add the new Phases set to scenario

21) Define objectives for your first phase. This phase will have two objectives: 1) Joe able to work on the spreadsheet; and 2) Joe working for a while without having the spreadsheet compromised by an attacker.

- Right click on the Objectives folder in the Reusable Sets Library pane. Select "new" and give the new set a name, "Phase1Objectives". Give the first element a name, "WorkSpreadSheet". Fill in the form as follows:
  - Set the phase to "0". The Objectives form names phases as integers starting at 0.
  - Completed Text: Joe is able to work on his spreadsheet
  - Uncompleted Text: Joe needs a computer with which to work on his spreadsheet.
- Click "Add" to add a second objective. Name it: WorkForTwoHours and leave the text fields blank except for Uncompleted Text:
  - Joe must work on the spreadsheet for an hour without having the spreadsheet compromised by an attacker. Check your physical security.
  - Set the phase field to "0".
- Note that we don't need "Completed" text for the last objective since we will move the game to phase 2 as soon as it is achieved. So there is nothing for the player to see.
- Add the Objective set to the scenario. Build and run. Click the "Objectives" button.

22) Define completeness criteria for the "WorkSpreadSheet" objective.

- Right click on the Triggers folder in the Reusable Sets Library pane. Select "new" and give the new set a name, "Objectives". Give the first element a name, "WorkSpreadSheet". Set the TriggerClass to "SetObjectiveStatus".
- In the Parameters, set the Objective value to "WorkSpreadSheet" (i.e., the name of the objective in the objective form. Leave the message blank.
- In "OneForMet", enter a "1". This sets the objective to "achieved" when the trigger fires.
- In the condition list, enter "NOT JoeFailsSpreadSheet". Thus, as soon as Joe no longer fails this goal, this objective will be met.
- Add this set to the scenario.

23) Define completeness criteria for the "WorkForTwoHours" objective. This objective is a little more subtle. This objective should not be achieved until the player has provided physical security to protect the asset. We could use the "ZoneHasSecurityValue" to measure the security of the zone. However that would not tell us if public has access to the zone. And, the player could choose to secure the entire zone. Or move Joe… In general, it is preferable to let the game engine do the work. So, we will simply measure whether the asset is compromised – or more accurately, not compromised over some period of time. If the asset is compromised, the player can still achieve this objective, but only after the asset is protected and Joe is able to work for an hour or so. We will define a condition to let us determine if the spreadsheet has been compromised.

- Right click on the Conditions folder in the Reusable Sets Library pane. Select "new" and give the new set a name, "AssetAttacked". Give the first element a name, "SpreadSheetAny". Fill in the form as follows:
    - Select the "AssetAttacked" condition type
    - Select "Joe's spreadsheet as the asset
    - Enter an "attack type" of -1 to mean "any attack type"
    - Provide a motive range of 0 to 1000.
    - Add the condition set to the scenario
- Select the "Trigger: Objectives" tab and click the "Add" button, providing "WorkForTwoHours" as the name of the trigger. Fill in the form as follows:
    - Select the "SetObjectiveStatus" trigger type
    - Provide a fixed delay of 0.1 (about 2 hours) Note the conditions that we define below will have to remain true over the course of two hours for this trigger to go off.
    - Set the Objective to "WorkForTwoHours".
    - Set "OneForMet" to 1.
    - Set the condition to: "NOT SpreadSheetAny AND_NOT JoeFailsSpreadSheet". (Note there is an underscore between the "AND" and the "NOT". The "AND" and "NOT" operators and the "OR" and "NOT" may not be used in conjunction, you must use "AND_NOT" or "OR_NOT".

24) Define conditions to measure completed objectives.

- Right click on the Conditions folder in the Reusable Sets Library pane. Select "new" and give the new set a name, "Objectives". Give the first element a name, "WorkSpreadSheet". Fill in the form as follows:
    - Select ObjectiveCompleted as the condition type
    - Select WorkSpreadSheet as the condition.
- Click Add and provide a condition name of "WorkForTwoHours"
    - Select WorkForTwoHours as the objective
- Add the new condition set to your scenario

25) Transition to the second phase; and delay Joe's Internet goal to this phase. Select the "Trigger: Objectives" form and click "Add", provide a name of "GoPhase2" and fill in the form as follows:

- Select the "SetPhase" trigger type.
- Enter 999 as the frequency and 0 as the delays.
- Enter "Phase Two" as the name of the phase to set.
- Provide text for a pop-up message, e.g., "Congratulations, you let Joe work safely for a couple hours! Now review your new objectives."
- Enter: "WorkSpreadSheet AND WorkForTwoHours" as the conditions.
- Note this is what was meant earlier when it was suggested that phases only transition based on ObjectiveCompleted conditions. We can now change the meaning of the different objectives without also having to change the conditions under which phases transition.
- Change the "Add Joe Internet" trigger to go off after the transition to phase 2.
    - Select the "Condition: Objectives" tab and click add, naming the new condition: "InPhase2"
    - Select "PhaseCompleted" as the condition class.
    - Select "Phase One" as the phase.
    - Open the "Trigger: Set Goals" set. Change the conditions to read: "InPhase2".
    - And set the trigger to "runs while paused". This will reduce the potential delay between phase transition and Joe's new goal.

26) Define Phase 2 Objectives. Right click on the Objectives folder in the Reusable Sets Library pane. Select "new" and give the new set a name, "Phase2Objectives". Give the first element a name, "AccessInternet". Fill in the form as follows:

- Set the "Phase From" value to 1 (which is the second phase)
- Set the display name to: Access Internet
- Set the uncompleted text to "get joe on the internet"
- Add the set to your scenario.

27) Play the new scenario. Build and run. Confirm that the second phase is not entered until Joe has worked a couple hours safely. When phase 2 is entered, confirm Joe has the new goal.

28) Add win and lose triggers. Open the "Conditions: Goals" set and click "Add" give the new element the name "JoeFailsInternet" and change the Asset Goal to "ReadWebPage". Click "Save". Then,

- Right click on the Triggers folder in the Reusable Sets Library pane. Select "new" and give the new set a name, "WinLose". Give the first element a name, "Win". Fill in the form as follows:
    - Set the trigger class to "WinTrigger"
    - Set the fixed delay to 0.1 and the random delay to 0
    - Enter text to display on the debriefing window: "Congratulations…"
    - As the conditions, enter: "InPhase2 AND_NOT JoeFailsInternet"
- Open the "Condition: Time" set and click "Add". Give the new element the name "Time8Hours". Enter "8" as the elapsed time, and "1" for "OneIfPerPhase". This means "8 hours have passed in the current phase".
- Select the "Trigger: WinLose" and click "Add" to add a lose trigger. Give it the name "LoseTimeOut". Fill in the form as follows:
    - Set the trigger class to "LoseTrigger"
    - Set the delays to 0
    - Enter text to display on the debriefing window: "Ooops…"
    - As the conditions, enter: "Time8Hours"
- Add the new set to the scenario

Scenario Development Tool Forms

This section describes the fields of the various SDT forms. Each form defines one of the SDT elements (e.g., users, physical components, goals, etc.) Many form fields define initial values that can then be altered by the player during a game. In these cases, the designer is directed to the CyberCIEGE encyclopedia for descriptions of these fields.

Scenario Form

Organization Name: Appears at the top of the game screen window.

Title: The scenario identifier in the Campaign Manager and Campaign Player.

Start Up Money: The amount of cash the player starts with.

Budget: Fixed amount of money the player receives per month.

Start Date: Day and time at the start of the scenario

Productivity Factor: This value is intended to refine the overall Bonus or Penalty experienced by the player. See the “Contribution” field in the user form. The intent is to permit fine tuning of a scenario without having to change each user’s contribution value.

Site Name: The name of the network domain associated with components within all the zones within the scenario, with the exception of offsite zones for which a distinct “Site Name” has been defined (see the Zone Form). Note that user domains are based on DAC groups as noted in the User form.

Site Description: Not used in game

Office Type: Selects art work for main office and offsite offices. There can be up to four offsite offices. The order of these selections corresponds to the order of offsite zones defined in the zone form. In general, add 30 to the Y zone corners to move from one offsite zone to the next.

Internet Name: The name of the Internet network, defaults to “Internet”.

Tutorial Attacks: not used; leave unchecked.

Quit Text: Text displayed when player quits.

Initial Briefing: Displayed on startup screen. Use “(PARAGRAPH)” to begin a new paragraph.

Full Briefing: Displayed in Game screen. Use "(PARAGRAPH)" to begin a new paragraph. Sometimes the gui cuts off the end of a briefing, so you may want to end each briefing text with (PARAGRAPH) to prevent truncated text.

Extra Syntax: Currently used to manage viewpoints, which are a set of camera positions that are cycled through via the tab key during game play. This is intended to provide easy component and user placement. See the section on "Generating Camera Viewpoints".

Camera Position: Initial game camera position. See examples for syntax and semantics. This is ignored if viewpoints are provided as described above, in which case the first viewpoint is used as the initial camera position.

Attack Masks: Attack types the will be masked at the start of the game. See the section on "Attack Types". NOTE: The masks are intended to be all set so that only triggered attacks occur. Non-triggered attacks will eventually be phased out.

Easy ACLs: If set, then selecting "Protect with ACL" for a component will cause the ACLs on that component to automatically match the intended access of the asset. If not set, the player will have to manually set the ACLs on the component if "Protect with ACL" is not set.

Guards Cost at Startup: If set, hiring a guard will cost one months salary up front.

Default WS Accounts: assigning user to a workspace having a computer automatically gives the user a local account on the computer.

NonServer Default Public: If set, then workstations purchased by players that are not assigned to users will default to local public access.

Attack Tickers: If set, game engine-generated tickers will scroll when assets are attacked.

Internet: Whether the scenario has an Internet network (i.e., something available to the public).

AssetTickers: If set, tickers announce user creation of assets.
Hide Unused Assets: Whenever assets are not part of non-zero goals, hide the asset. NOTE: these assets are not attacked, even though they may sit on computers.

Networks Everywhere: If set, then if a network exists in more than one zone it is assumed it can be wire-tapped from any zone. Note this is not affected by the zone "excluded networks" setting – though perhaps it should be.

Easy Training: Purchases a lot more training per click.

Static Internet: Prevents player from connecting/disconnecting to/from the Internet.

Zone Access Equip: If set, and a zone has defined guard positions (see Zone Form), then user access to the zone is potentially limited by player selected zone policies (e.g., "Eye Scanner". For example, if an eye scanner is required, then users are denied entry unless the eye scanner is present. User animations walk up to the checkpoint and perform an animation based on the type (or lack) of equipment.

Hide Unused Assets: Assets that are not part of some goal having a non-zero asset usage are hidden from the player.

User WS Names: Change workstation names to that of the user, e.g., "tom's workstation" when user is assigned to the workstation. This is useful when users are replaced via changes in asset usage, but note the designer supplied workstation name will be replaced if a user appears after startup. Names of static workstations are not changed.

No Loans: I f checked, the game prevents player from buying stuff if cash is zero or less. However, the designer can use the Loan trigger to enable loans to replenish capital lost to attacks or failed goals, (thus the field name is a bit misleading). If unchecked, then the player can lose & spend as much money as the scenario permits.

Lan PW Hash: Remote authentication and authentication servers are hamstrung such that they must put password hashes on the LAN in clear text. This forces players to select harsher password policies.

Hide Users: Hide users who have goals and a current total asset usage of zero.

Hide Night: Prevents users from going home at night, and keeps lighting from changing.

Force Camera: The camera will always return to the most recent user that it was set to (by a trigger). The "Patches" scenario is an example. It can be awkward because player has no ability to pan while unpaused. Also, the camera angles can be very tricky.

Simple VPNs: if selected, the VPN configuration dialogs don't require the player to select between public key and symmetric key encryption; and CA's and installed roots don't come into play.

Custom: Controls appearance of "custom configuration" entries in computer and device menus. Selection of these items (or double clicking a computer) will invoke the "doComponent" method of the Java "customClass". The returned string is processed as the name of a log file containing entries that are processed per the replay log processing described in "Debugging, Game Logs and Replaying Scenarios". The intent is to let developers create alternate interfaces for modifying the configuration of components.

Foreign: If not checked, users will not appear within component add-lists unless the user belongs to the same domain as the component. A user is considered to be part of the enterprise domain unless the user's default group has an associated domain that is not the enterprise domain, or the user is assigned to an offsite zone with a distinct domain name.

Need IDS: If checked, the player must buy an IDS and connect it to a network in order to see attack logs, network scans, and detect malware signatures on networks. See the "Install Malware" trigger.

CASA: If checked, the Attack Log button is labeled CASA, and computers can be turned off.

Network Form

Network Name: The name as it will appear in the NETWORK screen.

Network IP: not used

Static: Whether players can connect/disconnect components on this network.

Department Form

Users can be assigned to a department. There is no semantics to this. Note when defining support staff, the department is either SECURITY or TECH. Do not define these departments.

Name: The name of the department.

Zone Form

Distinct physically securable zones. This includes default procedural settings that will will be applied to computer purchased for the zone, and it includes physical security settings. Note users potentially become unhappy with some settings (e.g., cameras). If you set an enterprise user's "unhappiness immunity", the user will be immune to such things.

Name: The name of the zone as it appears in the scenario.

Zone Description: Text that appears in the ZONE screen.

Site Name: Only used with offsite zones, this defines the network domain of the zone. It may be the main office site name, which means it is part of the main office domain, or it may be a separate domain. Domains can have one level of nesting. For example, if the main office has a domain name of "blogo", an offsite zone can be assigned as a sub-domain of that, e.g., "subsidiary.blogo". A name of "Hide" will hide the domain and the hosts in dialogs that name remote hosts (e.g., filter exceptions). When displaying components in a "Hide" domain, the domain will be identified as "Conreel", which is an ISP. If the site name is "www", then each component within the zone is expected to have its own explicit domain name.

Static: Whether the player can alter attributes of this zone, including adding components.

Static Selectable: Like Static (above), but the physical security of the zone is based on selected properties. Otherwise, static zones have ultimate physical security.

Art: name of a .tga file that will be displayed in the ZONE screen when this zone is selected.

Zone Upper Left Corner…: The four corners of the zone in grid coordinates as displayed in the game via the "g" key. Zones are rectangles. When defining offsite zones, look at existing examples. In general, add 30 to the Y value to move from one offsite zone to another. See the Scenario form for identifying the type of office to use for each offsite zone. Define all offsize zones in the same scenario element set to control their order, which must match the order of the offsite zones in the Scenario Form.

> The first offsite office would typically have an upper left corner of 94, 29 and a lower right corner of 106, 21. The next offsite would simply add 30 to the Y values.

Procedural Settings: Name of procedural setting scenario descriptor. These are defined under the "Other" folder. Per the encyclopedia, these are default values applied to computers that are placed in this zone by the players. Note that once a user has been given procedural security instructions (i.e., been assigned a computer) that users settings will be inherited by any computer subsequently assigned to that user (e.g., if the users computer is stolen and then replaced.)

Guard Faces: If a guard is at the door to this zone, the guard will face in this direction. (Note directions are not consistent, use trial and error to get the guard facing in the proper direction, or reference existing scenarios.)

Guard X / Y: Location of the guard at the door in grid coordinates per the "g" key. If the "Zone Access Equip" switch in the Scenario Form is selected, then setting these values define the location of a zone access checkpoint. These may contain zone access equipment (e.g., eye scanners). Note the workspace from must contain an entry having the corresponding coordinates, and of the type "C" to complete the definition of a zone access checkpoint. The zone entries determine the location of the guard and the guard post (currently a small tall table). The workspace entries determine the location of the ID equipment and the spot at which characters will interact with the devices.

Order: Controls the order in which zones appear in the NETWORK screen. Set offsite zones to negative values (e.g., -1) to ensure they stay on the far right of the NETWORK screen. Avoid having offsite zones displayed in the middle of main office zones.

Queue: Intended for scenarios having physical zone checkpoints, this identifies the location where users while stand while they are waiting for another user to pass the checkpoint.

Constraints on User Access to the Zone: Per the encyclopedia "Physical Security" entry.

Excluded Networks: A list of networks that cannot be connected to computers that are in this zone.

Asset Goals: Intended for use in scenarios having "Zone Access Equip" in the Scenario Form to determine if the player has provided suitable zone access equipment and network connections. These goals are only achieved if an ID device (e.g., eye scanner) exists at the workspace that corresponds to the zone's guard position (Guard X/Y per above) and the device is connected to a computer that contains the target asset(s) and/or software. Assets will be automatically created on suitable computers if doing so will let the goal be achieved (e.g., used filtered software types and filtered assets to cause assets to be created on specific servers having specific types of software). If a zone has an ID Device and a corresponding zone access policy to use such a device, then authorized users are prevented from entering the zone **unless at least one** of the zone goals is achieved (not counting zone goals with the "ActivityLog" attribute). Additionally, the "ZoneFailsGoal" can be used to determine if suitable equipment, networks and filters are in place (e.g., to issue warnings before entry is denied). Also, if any of the zone's asset goals has the "visitor" attribute, then authorized visitors are blocked from entering the zone unless at least one such visitor goal is met.

- Note the game engine manages a set of assets from each active zone goal that has a filtered software type defined in the goal. This set is then used to list assets in a popup screen for configuring the ID device. If there is a single asset, then that asset will be selected by default. And if there is one asset that is an activity log that asset will also be selected. Otherwise, the player must select which assets will be accessed by the zone ID device. Zone goals will not be attempted (i.e., they will just appear as failed) unless the corresponding assets are selected. The intent is to let the player select between logging to a local database and/or logging to a centralized database (and between local ID databases and remote ID databases.)

Configuration Settings: Per the encyclopedia entry for "Components", these are configuration settings that are inherited by components that are placed in this zone. See the encyclopedia for details.

Secrecy Form

Secrecy Label: A text string to use as a secrecy label tag in this scenario file
For example, "Top Secret"

Level: A single value representing the hierarchical secrecy level between 0 and 64
This value has no semantics other than its relative value to other secrecy levels, i.e., it and the categories below determine the partial ordering.

Category: An enumeration of non-hierarchical secrecy categories, e.g., "A" or "A,B". Leave this blank if there are to be no categories.

Value of Secret: how much the player loses if an asset having this label is disclosed to an attacker.

Monthly Change…: not used

Value of Secret for an Attacker: The motive for an attacker to disclose assets having this label. Value is between 0 and 1000.

Initial Background Check: The degree of background checks initially applied to users who have this label as their secrecy clearance.

Integrity Form

Integrity Label: A text string to use as a integrity label tag in this scenario file
For example, “Accounting”

Level: A single value representing the hierarchical integrity level between 0 and 64
This value has no semantics other than its relative value to other integrity levels, i.e., it and the categories below determine the partial ordering.

Category: An enumeration of non-hierarchical integrity categories, e.g., “A” or “A,B”. Leave this blank if there are to be no categories.

Value of Integrity: how much the player loses if an asset having this label is modified by an attacker.

Monthly Change…: not used

Value of Integrity for an Attacker: The motive for an attacker to modify assets having this label. Value is between 0 and 1000.

Initial Background Check: The degree of background checks initially applied to users who have this label as their integrity clearance.

DAC Group Form

Users can be assigned to discretionary access control (DAC) groups, which can then be used to define "intended access" of an asset. For example: "all users in the engineering group should be able to read this asset."

Name: The name of the DAC group, e.g., "engineering".

Initial Background Check: The degree of background checks initially applied to users who belong to this group.

Asset Form

Assets can exist on a component at the start of a scenario. Alternately, assets can be created dynamically by virtual users based on the user goals. Attacks on assets are driven by the motives described below, and optionally by motives defined in the Goal Form.

Asset Name: The name as it appears in the game.

Secrecy: The secrecy label of this asset (if any). If secrecy labels are defined, then all assets must have some secrecy label. Disclosure of this asset has a cost and motive derived from the secrecy label in addition to those defined below under the cost list.

Integrity: The integrity label of this asset (if any). If integrity labels are defined, then all assets must have some integrity label. Modification of this asset has a cost and motive derived from the integrity label in addition to those defined below under the cost list.

Denial of Service Motive: Between 0 and 1000, determines the strength of denial of service attacks against this asset.

Availability Penalty: Monthly cost to the player if this asset is not available to users. Since it is monthly, this value must be large to have an effect.

Block User Assign: If set, users cannot re-assign this asset to other components. Only meaningful if the asset is initially on a component.

Create While Paused: If set, users will create this asset (if they can) while the game is paused. Otherwise users do not create assets unless the game is unpaused (i.e., to permit the player to establish the full network and security before users decide where to create assets).

HasDAC: keep this set.

Description: Displayed in the ASSET screen.

Intended Access Control List:

The intended discretionary access to the asset in terms of users or groups
of users and the specific access modes. The access modes are:
read, write, control and execute. Available modes are Y = Yes, N = No and X = Don't care or "don't care". A entry of:

joe YYYY

means that joe is intended to have read, write, control and execute access to the asset.
The entry:

joe YNNN

means that joe should be explicitly denied write, control and execute
access to the asset. There may be multiple entries. The most specific entry is applied to

each user and their desired mode of access. So, if Joe were within the Engineering group, the two entires:

joe YNNN
Engineering YYYY

would still deny Joe write, control and execute access. The "don't care" mode permits a users access to be whatever is specified for the groups to which the user belongs. For example, if Joe were in the Engineering group, then the two entries:

joe XNNN
Engineering YYYY

would mean that Joe should have read access to the asset.
By default, access is denied.

Cost List:

Defines the penalties incurred if a particular access violation occurs based on the "Intended Access Control List" defined above . In such an event, the "Cost List" is referenced to determine the cost based on which one of these groups or people access the asset. Name or group can be expressed as the wildcard character: "*". Mode is read, write, control, execute, or any.

Cost is the cash the player loses in the event of compromise.

Attacker motive is between 0 and 1000.

Mode for the cost list is, from left to right, read/write/control/execute where 'Y' triggers the property while 'N' does not. For example, YYNN would mean the that if read or write access is violated then the penalty amount in the cost field would be triggered; a NNYN would mean that only if the control of the asset is violated would the cost be triggered.

Actual ACL

Similar to the "intended ACL" described above, this is the actual initial ACL for the asset. Use this only if the initial ACL is to have some set value other than that defined in the intended ACL, and of course the asset must exist on some component at the start of the scenario.

Goal Form

Once defined, a goal can be given to one or more users. Goals are defined in terms of assets or/and software needed to access assets. Note also that zones can be assigned goals for purposes of testing access to identity databases and entry logs (see the Zone form).

Asset Goal Name: As it will appear in the USER screen.

Description: As it will appear in the USER screen.

Availability Penalty: Monthly cost to the player if this goal is not achieved by a user who has the goal.

Integrity Cost: A cost to the enterprise if the user's view of any assets in this goal is spoofed. Intended for use with read-only goals of high integrity assets.

Integrity Motive: The attacker's motive to present a user with a spoofed view of assets in this goal. Intended for use with read-only goals of high integrity assets.

Assets Defined for this Goal: If multiple assets are listed within an asset goal then it means that the user has to be able to access those assets at the same time.

> **Asset**: Name of the asset that must be accessed.
>
> **Filtered**: Whether user access to the asset could be filtered by a gateway component, and if so, what application service type must get through the filter. Filters that sit between the user and the target asset can potentially block the users ability to achieve the goal. The asset must be on the same component as filtered software. Filtered software types can potentially provide application level crypto protection, depending on the definition of the application per the Application Form. Some software types (e.g., Web Server) have SSL-based counterparts. The goal should specify the basic software type. When processing network filters, the game will adjust software types to their SSL counterparts if server settings require it. Similarly, some services (e.g., FTP) have SSH counterparts – and the goal should identify the service. The game will adjust the service to SSH if the client requires it. Note that in the CyberCIEGE software service abstraction, SSL can be prescribed per-asset for web; and for email it is either required or not. For SSH, there are no server-side settings prescribing use of SSH. That is controlled via the client settings (and player-defined filters, e.g., that might only permit SSH and not FTP).
>
> BackEnd: A non-blank value indicates the goal must access this asset as a back-end database using the specified software type. These are intended to be used in conjunction with "filtered" assets (per above) such that the filtered application on the component containing the filtered asset must be able to reach the back-end asset. If the back-end database is on a separate component, then the selected software type may be blocked by a network filter.

Read: Y means the user must be able to read the asset.

Write: Y means the user must be able to write the asset.

Execute and Control are not currently used.

Shared: if true then this goal will fail if any other goal having this asset as a shared asset fails. Similarly, if this goal fails, then it will cause all other goals having this asset as a shared asset to fail.

Goal is Shared: If set, then if multiple users are assigned this asset goal, they can only succeed in this goal if all users assigned this goal succeed.

Use Other Computers: If set, users will try to use other peoples computers to achieve this goal if they are unable to achieve the goal from their assigned computer.

PromiscDocs: If set, the user must access many external files that might contain macro viruses while achieving this goal. The effect is to increase the need for antivirus.

Visitor: Intended for use on Zone Goals (e.g., to limit user entry into a zone). If the zone has one or more goals that have the visitor attribute (but not also the "ActivityLog" attribute), than at least one of these goals must be met for "visitors" to gain entry to the zone.

ActivityLog: Intended for use on ZoneGoals to represent a need to log user entry. If "Log All Entry" is selected for a zone, and one of these goals is not met, the guard complains. This goal does not become operative until and unless the "Log All Entry" is selected. Also, these goals don't count as zone goals that determine if a user can enter a zone or not (see the Zone Form.) These zone goals don't become active (e.g., assets don't get created) until the "Log All Entry" is set. If the "Visitor" attribute is set, then ActivityLog is tied to both the "Log All Entry" and the "Scan All Visitors" zone parameters.

Software: Specific software applications that must be used when achieving this goal. Note the application can reside on any component to which the user has (potentially remote) access. Currently, prescribing specific software for SSL-related goals does not work – use software type prescriptions instead.

Software Type: Specific types of software applications that must be used when achieving this goal. Note the application can reside on any component to which the user has (potentially remote) access. Note that the "Messaging" or "Email Server" software types do not require users to have logical access to the associated asset (e.g., an ACL would not block achieving the goal) – only a suitable network connection is needed. Also note that the "Messaging" software type requires that all users having this as a shared goal must be using the same specific application of the Messaging type. Different applications of the same type are never "compatible" – the users must use the same application.

HarwareBase: A set of hardware base types. If not empty, then the user (or Zone User) must use one of these base types when achieving the goal. For example, a zone user goal might require use of a card reader. Note that if equipment is an ID type (e.g., card reader, scanner, etc.) and the user with the goal is a zone user, then the zone must also have the appropriate settings for the goal to be achieved (e.g., require iris scanner).

User Form

User Name: As it appears in the game.

Department: one of the departments defined above. This appears in the user description, but has no semantics unless its value is "Visitor" and "Zone Access Equip" is selected in the scenario form. Typically, a "VisitUser" trigger is used to cause the visitor to try to visit someone in the zone. Visitors do not appear in the user lists. Visitors can have goals. A Visitor can be given a DAC Group and/or a clearance that allows the visitor to enter the zone. These "authorized visitors" are prevented from a getting into the zone if the zone has a failed asset goal that has the "promiscuous docs" attribute (see Zone Form). Otherwise, such visitors are able to enter the zone. Unauthorized visitor (i.e., those whose group / clearance prevent them from entering the zone) can gain entry to the zone if "Permit Escorted Visitors" is selected.

Secrecy: The users secrecy clearance (if any). This determines the user's background checks, which effect the user's trustworthiness as defined below under "Trust".

Integrity: The users integrity clearance (if any). This determines the user's background checks, which effect the user's trustworthiness as defined below under "Trust".

DAC Groups: The DAC groups to which the user belongs. This potentially effects whether a user has "intended access" to an asset, and whether the user has a cost list motive (see the Asset form "Cost List) to compromise an asset.

Default DAC Group: This field is only used to identify the domain of users who belong to domains other than that of the main site. If this field is set, then the user's domain is the domain defined as part of the named DAC group.

Asset Goals: Identifies what assets the user needs to access as defined in the Goals form. Users may have multiple goals. Many users may have the same goal(s).

> Asset Goal: name of the goal per the Goals form.
>
> TargetUsage: A target usage of zero is the same as not assigning this user the goal. However, the target usage can be changed to a non-zero value via a trigger, thus introducing a "new" goal for user during the game. Target usage affects users' ability to share workstations to achieve goals. A workstation used by some user to achieve a goal with a high target usage is not available to other users. Note also, if a user has goals and the total asset usage is zero, and the scenario form has selected "Hide Users", the user will not appear in the game until a trigger provides a non-zero asset usage.
>
> Happiness: A value (integer between 0 and 100) is subtracted from the user's base happiness if the user fails the goal. Also see the efficiency algorithm described below.

Productivity: A value (integer between 0 and 100) is subtracted from the user's base productivity if the user fails the goal. Productivity can affect the player's Bonus / Penalty based on the user's Contribution and the "Productivity Factor" as defined in the Scenario Form. Also see the efficiency algorithm described below.

Efficiency algorithm:

The location of the component used by the user to achieve the goal can affect the happiness and productivity as follows:
Working on machine assigned to you = 0% change;
Working over network = 5% penalty
Working on any machine in your zone other than the machine assigned to you = 20% penalty;
Working on any machine not in your zone = 60% penalty.
Users cannot leave their current site to achieve a goal.

Trustworthiness: A value between 0 and 100 that contributes to how trustworthy the user is. In the simulation, this value is combined with background checks to compute a value between 0 and 900 as follows:

Background Check	User Trust
None	Trustworthiness
Low	50 + Trustworthiness
Medium	100 + 3 * Trustworthiness
High	400 + 5 * Trustworthiness

Also, Trustworthiness of greater than 95 will keep user from installing external software (i.e., "risky software"), regardless of procedural settings.

Initial Training: The user's initial amount of training (between 0 and 100). A value of 100 causes users to employ good password policies regardless of procedural settings.

Happiness: Initial Happiness level of the user

Productivity: Initial productivity of the user

Grid Position: An index into the workspace table identifying the user's assigned workspace at the beginning of the game. If the user has no assigned workspace, set this equal to the Start Grid value below. See the "Workspace Form". Note that these are workspace positions and not exact grid positions. For example, if a user is place into a workspace having no furniture, the user will be standing in a square that may not be the exact grid position, but will typically be within one grid.

Start Grid: Intended for users who do not have an assigned workspace and computer at the start of the game, and for users that are to start by standing at a location other than their workspace. This is an index into the workspace table. See the "Workspace Form".

Contribution: The amount of money this user contributes to the bonus / penalty calculation. This value is multiplied by the user's productivity percentage, and then by the Productivity Factor defined in the Scenario form (for fine tuning of the scenario).

Gender: male or female

Alternate Texture: Identifies an alternate animation to use when rendering the character. Values as follows:

0	maleworker2
1	maleworker2
2	maleworker2a
3	maleworker2b
4	maleworker2c
5	itguy
6	femworker3
7	armedguard_01
8	femworker2
9	alt_itguy
10	camoguard
11	femwoker2a (colored hair)
12	malewoker2a (plaid shirt)
13	maleworker2b
14	maleworker2c
15	femworker3

In the military environments

8 – female red blouse, black pants

Unhappiness Immunity: Whether user is immune to gross unhappiness resulting from cameras, armed guards, etc. In general it is suggested that this be set unless the designer builds in a lot of feedback to tell the player why the user is unhappy.

Wander: Whether user will wander if it has a workstation and a goal. **Warning**: offsite offices do not have doors, so if user's wander, they will go through walls.

Support Staff Form

Support staff are either technical support (who help keep components available and manage the component configuration settings the player selects) or security, e.g., guards. They can be working at the beginning of a scenario, or they can be made available for players to hire them.

Staff Name: as it appears in the IT Staff screen.

Department: Tech or Security (guards).

Harware Skills: 0 to 100, contributes to effectiveness of Tech staff

Software Skills: similar to Harware Skills

People Skills: Required to make other skills more effective, 0 to 100.

Available Date: Blank means working at the start of the scenario. "0" means available for hire immediately. Other values are the number of days into the scenario that the staff becomes available to be hired by the player.

Trustworthiness: Not currently used for support staff.

Initial Training: not used

Happiness: not used

Productivity: not used

Skill: How good a job a guard will do. 0 to 100.

Grid Position in Office: Index into the workspace table defining where the support staff will initially appear. Note that for guards assigned to doors (per the Zone form) they will appear at the location defined in the zone form.

Cost: Monthly cost the player pays for this support staff member. Note that if the Scenario form has "Guards Cost At Startup" selected, then this cost is also paid up front when the staff is hired.

Gender: All support staff are currently male.

Description: As it will appear in the IT Staff screen. Keep it brief.

Physical Component Form

These components exist at the beginning of a scenario.

Component Name: As it appears in the game.

Description: A description as it is to appear in the COMPONENT screen. Use the "(STATIC)" tag to designate the start of text that is also to be displayed in a message when the player tries to modify a static component.

Purchase Price: Not used.

Resale Value: Amount the player receives if the component is scrapped. But for pay-per-certificate Certification Authorities, this is the cost of each certificate. See publicCA below.

Maintenance Cost: not yet used.

Percentage of Time Available: initial component availability. This can decrease due to lack of IT support and malicious software.

Static: If set, the player cannot move, scrap, or alter the component configuration settings.

StaticConfig: Identical to static, but allows player to connect/disconnect networks. If checked, the Static setting is ignored.

NoScrap: If set, the player cannot move or scrap the component.

publicCA: If set, then this computer can be selected by players as a certification authority. The cost of these certificates is taken from the component "resale value". It is intended that such components are static and fully managed by the scenario designer.

Grid Position: integer index into the workspace table per the Workspace Form.

HwName: The hardware name displayed when this component is selected. If blank, then name is taken from the base component defined below.

Domain: The domain to which the component is assigned. Only meaningful for components in the "www" zone.

Assigned user: Initial assignment, if any.

Procedural Settings: Initial procedural settings for the component as defined in the encyclopedia entry for "Procedural Security". This is the name of a descriptor defined in the Others / Procedural folder.

Base Component: Determines the component properties such as whether it is a workstation or a server; whether it is a network device (e.g., router); or an identity management device. This also determines its shape in the office (unless it is placed in a server rack) and which operating systems are available for the component.

Operating System: See the encyclopedia entry for Operating Systems.

Software: Initial software installed on the component. See the encyclopedia entry for Applications.

Authentication Server List: List of authentication servers used by this component. The "Component Access List" of each of the authentication servers defines who can remotely access this component.

Network Connections: List of the names of networks as defined in the Network form, or connection names as defined in the “Others / Network Connections” form. The latter is used for connections that have ACLs or labels (i.e., for components having operating systems that enforce a MAC policy).

Assets: Initial set of assets on this component. Note an asset may only be on one component. If the “Hide Unused Assets” switch is set on the Scenario form, then the asset will not appear to the player in the asset list unless some user has a goal that includes the assest.

Local Access List: Per the Component Configuration Settings encyclopedia entry.

Configuration Settings: Per the Component Configuration Settings encyclopedia entry, except for:

> BrowserSettings:Loose on “Card Reader” devices implies the device simply reads a magnetic stripe, and thus is easily spoofed.

Link Encryptor Values: Applies only to components whose “Base Component” (defined above) is a link encryptor.

> Clear Text Network: The network over which clear text is sent and received
> Encrypted Network: The network over which encrypted text is sent and received.
> Key: The key of each link encrytor on either side of an encrypted network must match for information to properly flow between them. See the encyclopedia entry for Link Encryptors.

VPN Selection: the name of the VPN configuration element that defines the VPN connection profile and related settings. (See the VPN form.)

Catalogue Component Form:

These are components the player can purchase. Note also that components of the same type (e.g., servers) must be grouped together in the SDF, so use only one element set and arrange the order such that groups are contiguous.

Component Name: As it appears in the catalogue.

Purchase Price: Cost to player to buy the component.

Resale Value: Amount the player receives if the component is scrapped.

Maintenance Cost: If the sum of maintenance costs of all equipment (initial and purchased) exceeds 100, then at least one IT staff is required. Otherwise not visible to the player.

Percentage of Time Available: initial component availability. This can decrease due to lack of IT support and malicious software.

Base Component: Determines the component properties such as whether it is a workstation or a server; whether it is a network device (e.g., router); or an identity management device. This also determines its shape in the catalogue and in the office (unless it is placed in a server rack).

Operating System: See the encyclopedia entry for Operating Systems.

Software: Initial software installed on the component. See the encyclopedia entry for Applications.

Configuration Settings: Per the Component Configuration Settings encyclopedia entry, except for:

> Access Controls Fully Configured at Startup. If not set, then the ACLs on assets will be set to public.
>
> BrowserSettings:Loose on "Card Reader" devices implies the device simply reads a magnetic stripe, and thus is easily spoofed.

Also, for ID Devices, the "Browser Settings" reflect the strength of the ID mechanism with respect to ease with which an attacker can fool the mechanism. Note the O/S strength is also a potential weak link.

> Loose 20
> Normal 60
> Strong 999

Filter Form

Filters are used within gateway devices to block different kinds of applications from accessing assets through the gateway. This can affect a user's ability to achieve an asset goal, i.e., if the asset goal involves software that is blocked by the filter from reaching the asset. For each software type and network connection to the gateway component, traffic can be blocked in one or both directions. ***NOTE *** The direction is with respect to the network, which is different from the UI presented to the player. For example, "BlockOut" refers to traffic from the network (out of the network) to the gateway device. And "BlockIn" refers to traffic from the gateway device (in to the network). Each filter form only defines one "Filter Blocking Direction". Thus, if a given gateway is to initially block some applications in one direction, and other applications in other directions, then multiple filter forms for that same component and network connection must be defined.

Filter Name: Just a unique name. It does not appear in the game.

Filter Blocking Direction: To or from the networked named below. Note this can be confusing. If you want to block traffic coming into the site from the Internet, select the "Out" direction – it is traffic coming *out* of the internet wire into the gateway.

Component Device: the component that contains this filter.

Network: The network to which this filter is applied.

Applications Filtered: The filtered application types. See the Goals form for additional information on how filters can block user goals.

Workspace Form

The workspace form defines which areas of the office can contain desks and server racks. When playing the game, a player can only assign desks, computers, server racks and users to workspaces defined in this form. This form also permits the scenario designer to specify the initial content of a workspace, e.g., a desk or server rack. The columns are defined as follows:

Index
Corresponds to the "Grid Position" values assigned to users and computers in the "User" and "Physical Component" forms.

X and Y
Grid coordinate values that locate a workspace on the floor plan. These coordinate values can be viewed by running the game and pressing the "g" key and panning. Or you can simply refer to the floor plan figures in this document. Use values of zero to effectively remove an entry (without having to renumber all of the position indices).

Dir
The direction furniture will face. (N, S, E, W). Note directions are not accurate, i.e., when facing north, east is on the left. Use existing scenario workspaces as a reference and use trial/error to get it right.

Furn
The initial furniture as follows: A = desk; S = Server Rack; C = Zone access checkpoint; I = empty. The value "A" can be followed by two letters that dictate the optional workspace furniture for the left and right sides of the desk (respectively). These are L=lamp; S=shelf; F=cabinet ; T=trash can; N=none. If two letters do not follow the A, then optional furniture is random.

The refresh button reads the scenario elements and indicates which users and computers are currently assigned to each workspace. This is a developer convenience and it has no effect on the scenario. Note in the case of servers, only one computer and device is listed.

Warning: the "Remove" button can result in renumbering of workspaces but the tool does not update the corresponding user or component position indices.

Warning: if you define multiple workspaces close to the same grid position, users may wander unexpectedly. Use zero values for X to eliminate redundant workspaces (or delete them, but take care because the indices are not automatically renumbered.)

Condition Form

Conditions evaluate game state such as cash-on-hand, time, whether assets have been attacked, etc. Scenario designers can then cause "triggers" (defined in the Triggers Form) to fire based on as Boolean expression of the current game state.

Condition Name: The name that will be used in trigger condition lists. THESE MUST BE SINGLE WORDS WITH NO SPACES. Do not include the substring "_intrinsic".

Condition Class: The type of condition, e.g., UserFailsGoal, PhaseCompleted, etc. See the condition table below.

Parameter List: Values used to assess whether the condition is true or not. Reference the condition table below.

Note: condition values can be viewed while running a game from the SDT by pressing the "?" key. This requires the "-A" option in the Tools / Project Settings / Runtime parameters.

Trigger Form

Triggers permit scenario designers to cause specific game actions to occur in response to a given set of game state conditions. Triggered actions include popup messages, tickers, attacks, loss of money, etc.

Trigger Name: Unique name that will appear in the game log. Do not include the substring "_intrinsic".

Trigger Class: The type of trigger. See the trigger table below for details.

Frequency: The maximum frequency in days that this trigger might fire, e.g., "5" means once per 5 days. And 0.5 means twice a day. In the case of Attack Triggers having a motive of -3, this value also represents the period subsequent to the most recent asset compromise (or transition to a new phase) that this trigger's conditions will not be assessed[1]. See the section on "Managing Attacks". In all cases, after a trigger fires, its conditions will not be assessed until the frequency period has passed (less the delay values described below). If "runs while paused" (see below) is set, this value is real-time seconds. Attack feedback triggers that may occur more than once can be given a frequency of "-1" as a means of telling the engine to evaluate the trigger conditions after each asset compromise. This is strictly an optimization in lieu of setting a very high frequency, which could slow down the game engine.

Fixed Delay: A fixed number of days (expressed as a floating point number) after which a triggered event will occur. The trigger's conditions must remain true for this duration or the trigger will not fire. Also note that delays do not extend the frequency. For example, assuming the condition list remains true, a trigger with a frequency of 0.5 and a delay of 0.2 will fire twice a day. If "runs while paused" (see below) is set, this value is real-time seconds.

Frequency	Game minutes
0.1	144 (1 hour 24 minutes)
0.01	14 minutes
0.03	43 minutes
.042	~1 hour

Random Delay: A random number of days (expressed as a floating point number) that will be added to the Fixed Delay value to determine the period of time over which the trigger's conditions must be true. If "runs while paused" (see below) is set, this value is real-time seconds.

[1] The timing of the assessment of conditions is only of interest for conditions such as AssetAttacked which are true only once per trigger-per-event.

Runs While Paused: If set, the trigger's conditions will be evaluated even while the game is paused, and the trigger will go off if the conditions, frequency and delays are met. This is useful for help-tip trigger types.

Parameter List: A set of trigger-class dependent values that determine what the game will do when the trigger goes off. E.g., what message to display to the player; how much money to give the player, etc. See the trigger table below for details. When filling out text for display in text boxes, use "(PARAGRAPH)" to begin a new paragraph.

Slaves: Up to three triggers that will fire immediately subsequent to this trigger. Note the delays and condition lists of slaves are ignored. Also, some triggers will not fire their slaves unless the trigger "succeeds". An example is the VPNXCertCA trigger. Also note that slaves to message triggers won't fire until the message dialog is closed. The form includes a convenience listing of all triggers that are slaves to the trigger defined by the form.

Trigger Conditions

Unless a trigger is a slave, it will not fire unless its Boolean expression evaluates to true. The firing condition is express in terms of game conditions, and the form includes a convenience listing of all defined conditions for the scenario.

Condition List:

1) You can have up to 16 conditions in a trigger's condition list.
2) They can have up to 16 levels of depth of parenthesis.
3) Conditions must be connected by one of the following logical connectors : "and", "or", "and_not", "or_not"
4) The first condition can have a logical "not" in front of it.
5) Left parenthesis can only be placed directly in front of condition Names (no spaces), right parentheses can only be places directly after condition Names (no spaces).

Here are some examples.

true if and only if ShortTimeCondition is true
ConditionList: ShortTimeCondition :end

true if and only if ShortTimeCondition is false
    not ShortTimeCondition

true if and only if ShortTimeCondition and PlayerBroke are both true
    ShortTimeCondition and PlayerBroke

true if and only if at least one of ShortTimeCondition or PlayerBroke is true
    ShortTimeCondition or PlayerBroke

true if and only if ShortTimeCondition is true and PlayerBroke is false
ShortTimeCondition and_not PlayerBroke

Use of parentheses for nesting are illustrated below:
not (ShortTimeCondition or PlayerBroke)

not ShortTimeCondition or (PlayerBroke and KenHappy)

not (ShortTimeCondition or (PlayerBroke and_not KenHappy))

Dynamically Assessed Condition Values

Some conditions have values that can be assessed dynamically. And these "trigger condition values" can be provided by the designer as part of the trigger definition over-ridding whatever is specified as part of the condition definition. For example, you can define a "GameOnScreen" condition called "onscreen" and ignore its value parameter in the condition definition. You can then use this condition within a trigger's condition list, specifying the desired value in the "Condition values" section of the trigger form. For example, if a condition list read:
Time1hour and onscreen

And the condition value #2 was set to "2", then the "onscreen" condition would evaluate to true only if the player was viewing the office screen.

The criteria for evaluating condition values is usually a simple "equals". The following conditions evaluate as true if the game value is "greater than" the specified value:

- ASSET_TO_NETWORK_FILTER_COUNT
- TRIGGER_GONE_OFF
- ASSET_TO_INTERNET_NO_VPN

Most trigger condition values are integers. Exceptions are Register condition values which are typically "y", "n", "b", etc.

An alternate syntax for expressing a trigger condition value is to place an integer in brackets. This syntax is interpreted as 2 raised to the given power. E.g., "[7]" will be evaluated as two to the seventh, or 128. This is useful when expressing attack properties (e.g., "weak password"), which are bit masks. Multiple values can be strung together, e.g., [7][9], and these are ORed and then ANDed with the evaluated condition value.

Intrinsic Conditions

The TRIGGER_GONE_OFF condition can be assessed without defining explicit conditions. Place the desired trigger name in brackets within the condition list, e.g.,:
Time1hour AND [helpTip1]

Use of these intrinsic conditions requires use of condition values. In the above example, a condition value of zero would be true if the trigger named helpTip1 has gone off at least once. Intrinsic conditions can be qualified by time periods **expressed in real-time**

seconds. If the above example was changed to [helpTip1{5}], the condition would only be true five or more seconds after helpTip1 trigger had gone off. The intended use of that kind of intrinsic delay is to force a minimal spacing between messages to the user, e.g., to keep a user speak balloon from being overtaken by a pop up message or other event. Alternately, a time window during which the condition may be evaluated as true can be expressed as a negative value. In the previous example, [helpTip1{-5}] would be true only within 5 seconds of the helpTip1 trigger having gone off. Use two intrinsic conditions to express a combination delay and time window.

Intrinsic conditions may also compare the number of times a named trigger has gone off to the number of times the current trigger has gone off. A condition value of -2 causes the condition to evaluate to true only if the named trigger has gone off at least as many times as the trigger being defined. This serves as a shorthand for the CompareTriggerCounts condition.

Effects of Measuring on Condition State

Most conditions simply reflect game state, and are not tied to specific events. And most condition values are not affected by their being measured. Exceptions to this include the AssetAttacked and the CheckPointFailure conditions. The AssetAttacked condition will be true within a given trigger only once per attack event. If the same AssetAttacked condition is assessed as part of multiple triggers, it will be true once for each of the distinct triggers for each asset compromise. Similarly, the CheckPointFailure condition will only assess to a success or failure value once per trigger per zone entry event. Subsequent assessments of the CheckpointFail condition within that trigger will assess to the reset value (i.e., zero) until a new zone entry event occurs. All other conditions will evaluate to the same value multiple times to the same value so long as the game state remains the same.

Erasing Trigger Effects

Some triggers, e.g., speaks, can be configured to cause visible effects of the trigger to be erased if the trigger's condition list evaluates to false. For example, a trigger might cause a character to complain about a failed goal, and the resulting speak bubble will be erased once the goal is met. This behavior lets the designer avoid confusing the player with feedback that is not consistent with current game state. These types of triggers will not fire if their respective effects are still visible.

NOTE ON TRIGGER TIMING & SEQUENCING:

Triggers are generally assessed in the order in which they are generated in the tool (note that sets are organized alphabetically.) An exception is that help tip triggers and speaks triggers are assessed last. Triggers are not generally assessed while time is suspended (e.g., a popup is active, a screen other than OFFICE is up, or a pausing speaks is displayed.) Exceptions are help tips; "set objective status" and log triggers. Help tips are assessed last so that a trigger such as "set objective status" can affect a condition such that the help tip is blocked. As an example, consider a help tip displayed in the zone screen if physical security is too weak. The help tip condition could be based on an objective status. We want to assess the conditions that might set the objective status such that the help tip is suppressed if the physical security has been properly set while in the ZONE screen.

Procedural Settings Form

These elements are referenced by physical components and zones.

Name: The named used within physical component forms and zone forms to reference this procedural setting.

See the procedural security encyclopedia entry for detailed information on the fields within this form.

Component Network Connection Form

These elements are referenced by physical component forms. They define non-trivial network connections, i.e., those that have initial DAC or MAC attributes. Simple network connections can be expressed directly in the physical component form without reference to one of these elements. Currently, the only supported DAC attributes are access control lists that name users or groups of users.

Connection Name: The name used within physical component forms to select this element.

Network: The name of the network that is connected to the component that references this element.

Discretionary Access Controls: See the Component Configuration Settings entry in the encyclopedia for details.

MAC Connection Settings: See the Component Configuration Settings entry in the encyclopedia for details.

Phase Form

Each scenario can be divided into multiple phases, each having objectives the player must achieve prior to moving to the next phase. Phases and objectives are used to structure scenarios to lead players through several steps. The SetPhase trigger is used to transition from one phase to the next based on designer selected conditions. Designers are encouraged to limit these conditions to "ObjectiveCompleted" conditions. The game engine itself does not automatically tie phase transitions to the completion of objectives.

NOTE: Phases are order-dependent. Designers are encouraged to place all phases of a scenario in the same scenario element set, in the order in which the phases are to occur in the game.

PhaseName: As it will appear in logs and the named used in the SetPhase trigger.

Display Name: As it appears at the top the Objective Screen.

Completed Text: Appended to Display name after the phase has completed.

Uncompleted Text: Appended to the Display name until the phase has completed.

Objective Form

Each phase has one or more objectives; and a single objective can exist on multiple phases. The objectives of a given phase are displayed on the Objectives screen. Remaining objectives appear on the right hand side of the screen. Completed objectives appear on the left hand side of the screen. The designer entirely controls whether or not an objective is "complete" using the SetObjectiveStatus trigger type. The ObjectiveCompleted condition can then be used by the designer to assess the current state of any given objective. Designers are encouraged to use ObjectiveCompleted conditions when transitioning between phases.

Objective Name: Used in triggers and conditions.

Phase: identifies the phase(s) that this objective is part of. Integers between 0 (the first phase) and n where n-1 is the number of defined phases in this scenario.

Objective Completed: If selected, the objective is initially "completed". This can be useful for objectives that are to appear in the middle of a phase based on some player action. To achieve this, leave the "Display Name" and the "Completed Text" blank.

Display Name: The first part of the objective description on either side of the screen. Leave it blank and just use the "Completed Text" and the "Uncompleted Text" to provide a cleaner presentation to the player.

Completed Text: Displayed on the left side of the screen if the objective is achieved.

Uncompleted Text: Displayed on the right side of the screen until the objective is achieved.

Application Form

If no application software is defined, the game engine defaults are used.

Application Name: the name as it appears in the game.

Application Type: See the definition of software types in the Goal Form

Cost: What it costs the player to install the software. A negative value will prevent the software from showing up in the buy list, while allowing it to be preinstalled on catalogue components.

Patches: how frequently the application needs to be patched.

Assurance: The likelihood of the application including malicious software or exploitable flaws. Intended for use in integrity scenarios. Also, see "Crypto" below. The application's assurance can change during a game based on exposure to the internet and malicious software. In general, assurance is halved if the software serves the internet and patch requirements are worst than "few"; or if malicious software is present on the component.

Possible O/S: Operating systems that can run this software

Description: As it will appear in the game.

Crypto: Whether the application is crypo-enabled. Use of crypto-enabled applications to achieve shared goals in peer-to-peer configurations -- or isolated goals in client-server configurations -- can protect the asset from wiretaps, up to the assurance noted above. The designer must define the goal such that the software type that might provide crypto is "filtered", otherwise no protection is provided. If all filtered software used to achieve the goal is crypto-enabled, then protection is provided up the least assurance of all such software. Note that SSL-enabled applications are managed separately and should not have the crypto field enabled.

No One-time passwords: Intended for use with software services (e.g., web servers.) If true, the application is unable to support one-time password protocols. If the player selects one-time passwords for a server having a filtered service used in a goal, the goal will fail advising the player of the incompatibility. This switch is intended to force players to use TLS protocols, or some other means of remote endpoint authentication in environments where passwords might be compromised.

VPN Form

This form describes VPN parameters that are then referenced by component forms. The form is currently just a button that, when clicked, will display the very same VPN configuration dialog that players use in the game. See the CyberCIEGE encyclopedia for definitions of those field values. Note that while these dialogs within the game will hide irrelevant fields from the user (e.g., Local LAN fields when configuring a client), the SDT display of these dialogs does not hide unused fields.

Within the game, when a player selects a symmetric key ID that is being used remotely, the player is assessed a cost for exchanging keys. However, any initially configured key Ids on static components will not incur this cost. The intent is to let the designer lure the player into using a key that is already in use somewhere in the enterprise (e.g., by a rogue user).

The "Simple VPNs" switch in the Scenario Form determines whether the scenario will require players to choose between public key and shared secret keys for the VPNs.

Pay-per-cert CAs are distinguished by the "publicCA" switch in the Physical Component form. The cost to the player is taken from the component's resale value. It is assumed that these components will be static and fully managed by the scenario designer.

The costs of distributing shared secret keys are currently fixed as follows:

Second and subsequent use of key locally: $500

First use of key previously used at remote site in the same top level domain: $1500

First use of key previously used at remote site of a different top level domain: $3000

Cross certification of certificates is represented by defining the signer as an installed root within the subject CA's VPN definition. Players see this by looking at what CA certs a given CA has signed (i.e., the player does not see the VPN dialog for a CA).

Email Form

This form describes email parameters that are then referenced by component forms (e.g., workstations). Note the form is primarily just a button that, when clicked, will display the very same Email configuration dialog that players use in the game. See the CyberCIEGE encyclopedia for definitions of those field values. The form also includes lists for selecting which users' email clients are subject to the configuration defined by the form.

Email assets are intended to be per sender/receiver/subject. Joe's email to Fred about foo bars is a separate asset than is Fred's email to Joe about foo bars.

Email goals are defined by including Email Server type software and Email Client type software in the goal definition. The Email Server type software must be defined as a filtered software type (see the Goal Form). The receiver of the email should be given a read-only goal, and the sender given a modify goal to the email. The email asset is distinguished by being the "filtered" asset within the goal (and it is assumed such goals have only one filtered asset.)

If the criteria for sending and receiving email are met (see the email encyclopedia entry) then the sender will create the email asset upon the receiver's email server.

Email assets are intended to persist on the recipient's email server. The assets themselves do not move to the recipient's workstation. However the attack engine will temporarily move protected email assets to the sender and receiver's workstations to determine if malicious software could compromise the asset.

Attacks on email will succeed or fail based on current settings. Thus, while users could exchange unencrypted emails prior to the player configuring clients for encryption, the emails would be protected as if they had been encrypted. The same holds for signing email. If the asset goals associated with an email go to zero usage, the asset's protections remain in their most recent states.

When considering attacks on email assets, we look for suitably configured sender and receivers and offer protection to the asset even though the asset itself might be compromised. The assurance is based on the assurance of the email clients and PKI CAs if any. Protection is provided for in-place assets and wiretaps, however Trojan horses and attackers who succeed in authenticating themselves as the authorized user can compromise assets. Attacks on protected email start with attempts to compromise the key. If that is on smart cards, then the email assets are effectively moved to the sender's and the receiver's workstations to see if they can be compromised there.

If an AssetUsage trigger causes a goal to become null, the last settings in place when the goal was achieved are used to determine if the keys can be compromised, and thus the asset. Note though that attacks on the asset at the sender and receiver client will cease.

The sender of an email will encrypt or sign the email only if directed to do so. This assumes training is at least 70.

An email receiver will insist on email being signed directed to do so.

When defining email assets, set the "intended access" to reflect RW access for the sender and R access for the receiver. If you assign group based access, subtleties in the attack engine will keep malware on clients from compromising the asset.

If the player (or initial scenario) requires "CRL" for an email goal, then the workstation must be able to access a CRL server via the LDAP application service. A CRL server is any server containing an LDAP software type application.

SSL Form (Client and Server)

This form describes SSL parameters that are then referenced by component forms. Note the form is primarily just a button that, when clicked, will display the very same SSL Client or SSL Server configuration dialog that players use in the game. See the CyberCIEGE encyclopedia for definitions of those field values. The forms "Software Type" field identifies the kind of SSL-enabled software (e.g., web server, browser, etc.) The client form also includes lists for selecting which users' SSL-enabled clients are subject to the configuration defined by the form. An empty user set denotes the default client configuration.

Cross certification of certificates is represented by defining the signer as an installed root within the subject CA's VPN definition. Players see this by looking at what CA certs a given CA has signed (i.e., the player does not see the VPN dialog for a CA).

Question Form

The question form has been added to replace most uses of the Question Trigger (which required separate message triggers for each different user response.) Question forms are named by the QuestionMult trigger classes. Questions are designed to optionally permit looping of the question until the player selects the "correct" answer. The QuestionMult trigger class supports responses in the form of messages and user speaks.

Question: the text of the question as it is to be displayed to the player.

Type: Reflects whether this question is intended as a quiz question, or a guidance question. This value affects log entries and potentially tools that consume logs. Questions of type "quiz" are not asked again if the player has previously correctly answered the question, as reflected in the logs.

Answers: the possible answers as they are to be displayed to the player. Each answer will be displayed with the corresponding "A", "B", etc. as a button next to the answer.

Responses: Messages or speaks are displayed containing this text when the player selects the corresponding answer.

Answer: the correct answer to the question. This will be included in the log entry as the, simplifying student assessment. The log entry includes a response string whose first character is the correct answer, and subsequent characters are the player selections (more than one is possible if looping is enabled as per below.)

Loop: If selected, for a given answer, the question will again display after the response message is closed. Correct answers will not result in loops, regardless of the setting.

PacketXform Form

This form maps in-game component names to generic component names found within network traffic packet samples. These forms (and the network packet samples) are named by PacketXform triggers. The transformed packets are written into a packet log for the component named by the first table entry. Component names have two parts: the component name and the domain name. These may be configured to be random values as noted below. The table columns define the name transforms as follows:

Generic name: The name as it appears in the network packet sample, e.g., "Source".
Game name type: Component names that appear in the packet logs are selected in one of six ways:

> Goal: the name of a workstation used by a given user to achieve a given goal.
>> Game name: name of the goal
>> 2nd game name: name of the user
>
> Asset: the name of a computer that contains the given asset.
>> Game name: name of the asset
>> 2nd game name: not used
>
> Text: Explicitly provided names.
>> Game name: name of the component
>> 2nd game name: name of the domain
>
> Random: results in random component and domain, game name and 2nd game name are ignored.
>
> GoalRandom: similar to "Goal" above, but the component name is random (the domain name is per goal and user.)
>
> AssetRandom: similar to "Asset" above, but the component name is random (the domain name is per the asset).

If the packet log is to be collected at some component other than the source or destination, create a first table entry of Generic Name "sniff", of type "Text" and provide the component name as the Game name. NOTE: the game does not check network connectivity, i.e., packet logs might appear on components that are not on the same network as the source / destination. You must control this.

Advanced Scenario Development

This section describes strategies for developing scenarios.

Phases and Objectives

Non-trivial scenarios are generally organized into multiple phases, each of which includes one or more objectives that the player must achieve. Objectives and phases are the best way to help the player progress through a scenario without losing sight of what you (the designer) are trying to get across.

Mechanically, triggers and conditions are utilized to transition between objectives and phases. SetObjectiveStatus triggers control whether the game engine views any given objective as being completed. Similarly, the SetPhase trigger causes the game to enter a new phase. The firing of these triggers is entirely dependent on conditions defined by the designer. It is suggested that designers reference existing scenarios for examples of how objectives and phases are defined, and how transitions are achieved.

As an organizing rule of thumb, it is suggested that SetPhase triggers be defined only in terms of ObjectiveCompleted conditions. This generally makes it easier to maintain scenarios and to add new objectives to phases. The SetObjectiveStatus triggers are then defined in terms of game play state, i.e., conditions that reflect choices the player has made and consequences of those choices. For example, a "Protect Asset" objective might be explained to the player as a need to protect a specific asset while allowing user's to access the asset in pursuit of goals. An AssetAttacked condition could measure whether the asset has been compromised and a UserFailsGoal condition could measure whether the player has provided a user with resources necessary to achieve a goal. A combination of those two conditions and a fixed delay might then trigger a SetObjectiveStatus trigger that marks some objective as being completed.

Designers often find it helpful to add test jigs to scenarios under development in order to advance the scenario to a later phase without having to perform all the requisite steps. It is therefore a good idea to qualify the scenario's SetPhase trigger with an otherwise redundant AND_NOT [phase completed] condition. The purpose is to suppress the scenario's SetPhase triggers when test jigs are advancing the phases.

User Goals

The scenario designer defines user goals as a means toward forcing the player to deploy and network computers. When users fail their goals, they become unhappy, lose productivity and bang on the keyboard. The "Availability Penalty" is the easiest way to cause the player to lose money when a goal is not achieved.

Condition Assessment

A scenario designer's ability to "move the game along" depends in large part on the ability to assess game state and cause things to happen based on that state. Reference the "Trigger Form" description for details of the (sometimes subtle) syntax and semantics of game condition assessment. Most conditional state assessment is relatively straight forward. For example, it is easy to measure if the component containing some asset has a specific policy. And it is easy to assess whether a user is achieving a specific goal, and whether the user has achieved the goal for a given amount of time (see fixed and random delays in the trigger form). The question of whether a given asset has been attacked (i.e., the AssetAttacked condition) is a bit more subtle as detailed in the Trigger Form description. The same is true for the question of whether a virtual user has passed or failed a zone checkpoint (CheckpointFailure). The states of these two events are assessed from the perspective of the individual triggers whose conditions perform the assessment. Thus one trigger may assess an asset as having been compromised while another trigger at the "same time" sees the asset as having not been compromised because that trigger had already seen that particular instance of an asset compromise. As detailed in the Trigger Form, a single instance of an asset compromise will only be seen once per trigger[2]. The same is true for CheckpointFail conditions – they are assessed as non-zero only once per zone entry attempt per trigger.

If a simple measure of whether an asset has ever been attacked (or how many times the asset was attacked) is needed, then use the AssetAttackCount condition.

Measuring Time

The passing of time can be measured via the TimeCondition, which lets designers assess time in minutes or hours, either from the beginning of the scenario or from the beginning of the current phase. The "Fixed Delay" and "Random Delay" values also allow the designer to incorporate time into the decision to fire a trigger. Note that the semantics of these delays is quite different from the TimeCondition. The condition simply determines if a given amount of time has passed without regard to other game conditions. On the other hand, the delays allow the designer to prevent a trigger from firing until all the trigger's conditions have been met continually for the period specified by the delay.

Use of delays in triggers allows the designer to draw logical conclusions about the player's ability to protect assets without measuring individual component and network properties. The discussion above described an example "Protect Asset" objective that achieves this. Since the designer controls the frequency of AsssetAttack triggers, the designer can determine that the asset has indeed been protected from one or more attacks. Similarly, the designer can easily determine that the users have had uninterrupted access to the assets for a specified period of time. Thus, delays can be used to determine if game state conditions have persisted over some period of time during which some defined set of events have occurred.

[2] The AssetAttacked condition includes a "WithinSeconds" parameter that will hold the condition as true for a given number of game seconds after the asset compromise.

Note that if the trigger's condition evaluation changes to a false state during the specified delay period, the trigger will not fire. To cause a trigger to fire some fixed time after some game state has been reached (or some fixed time after the game state has persisted per a delay value), split the trigger into two triggers: the first fires when the conditions are met and is invisible to the player (e.g., is a LogTrigger); the second trigger would then include the first trigger's firing as an intrinsic condition (see the Trigger Form) and the second trigger's specified delay controls the period between the firing of the first trigger and the firing of the second[3].

Another means of accounting for time is to express a delay within the assessment of intrinsic TriggerGoneOff conditions as noted in the Triggers Form. Whereas a typical delay lets the designer say: fire this trigger when its condition expression has continually evaluated true for this period of time, the intrinsic condition delay lets the designer say: "fire this trigger when its condition set is true, but not within N seconds of that other trigger going off." Note that here the value of "N" is in real time seconds, while the standard delay is in terms of game time (with the exception of runs-while-paused triggers). Thus, intrinsic delays can be useful when you want to ensure a speaks or some other feedback is not over-written by new feedback too quickly. This is particularly important when the player compresses time.

Intrinsic conditions also allow the designer to specify a time window to say "fire this trigger when its condition set is true, but only within N seconds of that other trigger going off." The latter form is useful when you've split a trigger into two and you want the second half to fire only after each instance of the first half of the trigger fires. For example, if you give the second trigger a frequency of 0.1, and its only condition is, [thatTrigger], with a condition parameter value of zero, then the second trigger will fire every tenth of a day subsequent to the first firing of the "thatTrigger". But if you alter the condition to read [thatTrigger{-10}], the second trigger will only fire within ten seconds of the most recent firing of thatTrigger, thereby keeping the two triggers in step. This form of time constraint is also helpful to bind question responses to their questions in a way that lets the same response occur multiple times, assuming the question is re-asked.

Where Time Delays Don't Work

Using trigger delays to infer that zone entry attempts have occurred is not always practical because the designer has little control over the timing of zone entry attempts (e.g,. users may be queuing up at the zone checkpoint). This is due to the synchronous tie between the animated screen activity and the game engine events as described above in "User Animations and Interactions".

Using delays in triggers containing CheckpointFail conditions is not appropriate because such events are only measured once per event, i.e., they have no duration. Similarly, using delays in triggers that should fire when a given asset compromise has occurred is

[3] The trigger form and game processing could have been extended to include this kind of delay, however it is likely that designers would want to use two triggers anyway to permit special-case cancellation of the second trigger.

not appropriate because those events have no duration[4]. On the other hand, using delays in triggers that should fire when an asset compromise has NOT occurred for a period is appropriate because the lack of a compromise has a duration.

Combining multiple AssetAttacked and/or CheckpointFail conditions in one trigger is also not generally useful[5]. This is because the question being asked is not: "has that asset been compromised?"; rather, the question is "has that asset been compromised since that last time this trigger's conditions were evaluated?". And the frequency of trigger evaluation is not generally predictable (and you don't want to try to predict it.)

Note that the nature of the CheckpointFail condition (described above) lets the designer determine that a checkpoint entry attempt has occurred without relying on time. By combining a CheckpointFail condition with a ZoneGoal condition, the designer can measure whether some user has successfully passed a checkpoint while some zone goal was achieved. CheckpointFail condition is generally a better choice than the UserInZone condition when assessing whether user zone access succeeded while some conditions remained true.

Question Triggers

Designers use question triggers to test a player's understanding of material and to potentially alter the direction of a scenario. Questions can be either true/false or multiple choice. There are two forms of question triggers. The "Question" class of trigger assumes the scenario designer will create responses using other triggers. See the Question trigger table entry for detailed syntax of questions. The "QuestionMult" form of the trigger makes reference to a "Question" form that defines the question and the responses (i.e., pop-up messages to display in response to each player answer.) Both forms of question trigger are assigned what should be a unique register condition. As noted in the Register condition table entry, a register is simply a named state variable. Registers have an initial value of zero. When a question trigger fires, its associated register is assigned the value of the keyboard response provided by the player.

The QuestionMult triggers should be used when player feedback will be in the form of pop-up messages. The question forms greatly simplify question and response management for such questions, (relative to managing responses use with Question Triggers.) If the player feedback will not include pop-up messages, then Question triggers should be used. These require additional logic as is described further below.

QuestionMult triggers have two types, "Quiz" and "Guide". The former are intended for use in testing the player's knowledge. Once a player correctly answers a "quiz" question, the question is not asked again, even if the player restarts the scenario, (unless the player deletes the log.) Guide questions are intended to help prod the player back to the expected game path and understand specific concepts.

[4] Delays can be used with AssetAttacked conditions to provide a brief delay between an attack and feedback – but use that carefully to avoid feedback from vulnerabilities the player has already corrected.
[5] No doubt someone could find an exception to the general rule, though that would likely rely on assumptions about the trigger assessment sequencing and timing that might not persist.

When using “Question” triggers (but not “QuestionMult” triggers), designers trigger responses to player-provided answers based on the value of registers. For example, a user speaks trigger might cause a user to say: “correct!” when the question trigger’s register has a selected value, e.g., “b”. The response to an incorrect player answer can be uniquely defined for each potential incorrect answer. Or a single “wrong, …” response trigger can be designed to fire when the associated register value is NOT 0 and NOT “b”.

When providing responses to individual incorrect answers, it is sometimes desired to re-ask the question until the player selects the correct reply. One technique for achieving that is to set a high frequency (e.g., 0.02) for the question trigger and include a “NOT (correct answer)” condition. Each response to incorrect answers would also have a high frequency and would include an intrinsic delay on the question trigger that suppress the response five or so seconds after the question trigger fires, e.g., use an intrinsic condition such as “… AND [questionTrigger{-5}]”

Assessing Event Sequencing

Sometimes the designer needs to ask: “Has event A occurred since the most recent occurrence of event B ?” One example of such a question might be: “has Joe passed the checkpoint since the most recent compromise of Asset X?” The “EventSequence” condition can answer this question for events that include asset compromise (as reflected by AssetAttacked conditions), zone entry attempts (as reflected by CheckpointFail conditions) and the firing of triggers (as reflected by the TriggerGoneOff condition – either explicitly or via an intrinsic condition.) See the EventSequence entry in the trigger table below.

Managing Attacks

Scenario designers must manage the occurrence of attacks using triggers. The game engine determines the success of attacks based on attacker motive, network topology, configuration settings, procedural settings and user training. In general, it is best to let the game engine derive attacker motive from the individual asset motives (and thus the success of any given attack type) by using a value of “-2” or “-3” as the motive in the attack trigger. The game engine also gets attacker motives from Goal definitions, though these are only intended for use with goals having read-only access to high integrity assets.

When an attack trigger fires, all assets are attacked via the defined attack type by all attackers (including insiders where appropriate), in the modes prescribed by the asset’s motive values. Each asset will not be compromised more than one time as an effect of an attack trigger firing. Also, the value “-3” as a trigger motive value will cause the attacks on other assets (due to this same trigger firing) to cease subsequent to a compromise. A motive of -3 also delays the evaluation of an attack trigger following any asset compromise, and subsequent to a change in scenario phase. In other words, the attack trigger cannot possibly fire for a specified period. The delay period is taken from the trigger frequency (e.g., a frequency of 0.05 would ensure the trigger cannot possibly fire within 0.05 days of the most recent asset compromise.) Again, this semantics is selected by setting the trigger motive to -3.

Attack triggers cause attacks on each asset as described above. The order in which assets are attacked is based on the motive that an outside attacker has to compromise the asset in any mode. Distinct insider motives do not affect the order of attacks.

Often, a scenario designer will want to delay occurrence of attacks until some state is reached (e.g., until the player has achieved some basic objectives such as letting users achieve goals.) This can be done simply by including a UserFailsGoal condition in the attack trigger. Also, it is often desirable to delay the occurrence of some attack types until the player has managed to successfully defend against other attack types (e.g., delay insider attacks until the player protects against Internet based attacks). Since triggers within a given scenario element set are evaluated in order, the easiest way to sequence attacks is to sequence their triggers and use a motive of -3 as described above.

When an attack is successful, some components may retain attack properties that become visible when the player selects "diagnostics" from the component menu. However, these attack properties get reset whenever any new attack is attempted for any asset. Therefore if the designer wants the player to reliably see these attack property values without race conditions, then the attacks within any triggered cycle must cease immediately after any attack is successful. This can be done by including a "not" on an "any asset attacked" condition and setting each attack trigger's motive to -3 as described in the AttackTrigger description.[6] The motive of -3 sets the minimum delay before an attack is attempted, and the NOT AssetAttacked condition ensures that this trigger's attack type is the first out of the gate subsequent to the time delay.

In some cases it is helpful to initially set some asset motives very low (e.g., zero) and then increase the motive with a trigger when the scenario reaches a suitable point. This can prevent a player from being overwhelmed by attacks against a lot of different assets and can help the designer to focus the player's attention.

Responding to Attacks

When assets are compromised, the game engine generate tickers and displays animated dollar amounts unless the scenario form's "Attack Ticker" switch is off.

Designers often prefer to display customized messages or speaks in response to compromised assets (and thus turn off the Attack Ticker switch). It is relatively straight forward to provide the same response to each occurrence of a given compromise. The designer can create a fixed sequence of different responses by placing intrinsic TriggerGoneOff conditions within the different response triggers, combined with AssetAttacked conditions. For example, the second response trigger would include the

[6] Typically, the designer will expect this technique will suppress asset attacks until the trigger for the previously successful attack type comes around again on the trigger cycle. When assessing that same trigger subsequent to the attack, the engine will force the AssetAttacked condition to false so that the trigger can go off again without skipping a cycle and letting some other trigger get a chance to go off.

first response trigger as an intrinsic condition. The third response trigger would include the second, and so forth. In this strategy, the intrinsic conditions are what determine which of the triggers will fire.

Note that the use of ticker triggers in response to attacks is potentially confusing because race conditions might lead the player to only notice the ticker *after* the player had corrected the vulnerability that led to the attack. A message trigger is much less likely to result in such race conditions because they can occur immediately after the attack and prevent the player from doing anything until the message is acknowledged. Combining a message trigger with other feedback triggers (e.g., speaks or burning computers), slaves is one way to avoid potentially confusing the player while providing richer feedback than just popup messages. Attack feedback triggers that may occur more than once can be given a frequency of "-1" as a means of telling the engine to evaluate the trigger conditions after each asset compromise. This is an optimization for the game engine as an alternative to setting the frequency to a very small number. Note, a delay value combined with a frequency of -1 would cause the trigger to never fire.

Sometimes the designer prefers to create a random sequence of responses to compromised assets. This can be achieved using the RandomValue condition class. The designer creates a set of triggered responses, each of which includes the AssetAttacked condition as well as a unique (among the triggered responses) pick of the random value.

Random Values

This paragraph details how RandomValue conditions are assigned new random values. This is critical to a designer's ability to manage random responses to events. The RandomValue condition is only assigned a new value subsequent to the firing of a SetRandom trigger that names the condition. However, the game engine will delay the assignment of the new random value until after each *dependent trigger* has been assessed against the *latest recorded game state,* including the current random value of the condition. A "dependent trigger" is any trigger that includes that RandomValue condition in its condition list and is ready to fire with respect to its frequency. The "last recorded game state" is a per-condition state variable that is set each time the condition's value changes, and (more importantly for many applications) each time a dependent trigger fires. For example, consider several triggers that are dependent on their own unique values of a RandomValue condition along with an AssetAttacked condition. And each of these triggers has a slave SetRandom trigger to change the random value after the trigger fires. When the asset is compromised one of the triggers will fire per its selected random value (i.e., the value of the RandomValue condition at the time of the asset compromise), and the slave will then fire to change the random value for the next round. However, before the random value is actually changed, each of the other dependent triggers are guaranteed to be assessed against the current random value, thereby allowing each of the other dependent triggers to "see" that AssetAttacked condition is true such that those other triggers will only potentially take notice when some subsequent attack succeeds.

Fatal Attacks vs Survivable Attacks

When the consequences of attacks are relatively mild, scenarios play reasonably well by letting attacks occur and providing players with feedback about what happened and how to prevent it from happening again. However, when consequences of attacks are to be severe or catastrophic, continued play of that scenario might become difficult to manage. For example, a large loss of money might leave the player without funds to continue. Or, continuing after the loss of a critical secret formula might be too far outside the scope of the scenario's narrative.

A general rule of thumb for creating scenarios is that when severe losses occur, the scenario should terminate with a loss and the player should then have the ability to restart the scenario at the end of the previous phase before the large loss occurred.

Warning Players About Vulnerabilities

When large losses might result from an attack, the scenario designer should give the player warnings about vulnerabilities that exist before attacks are generated. Such warnings should be given at least within scenarios that first introduce concepts associated with distinct vulnerabilities. Similarly, it is often helpful for the player to receive warnings about vulnerabilities that could lead to survivable attacks.

Ticker triggers are one way to warn players about vulnerabilities (regardless of the extent of the consequential loss). The trigger table entry for ticker triggers describes how to cause a ticker to be withdrawn when the trigger's conditions are no longer true. This avoids the potentially confusing situation in which the player fixes a vulnerability, but continues to see the warning ticker scroll across the screen. The designer should take care to ensure that the tickers will be withdrawn (i.e., erased) if the player takes appropriate action. If a ticker trigger definition flags the ticker as a candidate for withdrawal, then subsequent to its firing, the game engine will continually reassess the trigger's conditions. If the conditions evaluate to false, the ticker will be withdrawn. Note re-evaluation of conditions for ticker withdrawal is not meaningful for AssetAttacked or CheckPointFail conditions because those are only true once per trigger per event. Therefore, when evaluating a trigger's conditions to see if a ticker should be withdrawn, the game engine holds those two condition types as "true". Also note that issuing ticker warnings based solely on vulnerability assessment conditions such as "AssetToNetwork" can be challenging since the designer may wish to withhold warnings about some vulnerabilities until the player has mastered other vulnerabilities. One useful strategy is to predicate vulnerability warnings on a combination of AssetToNetwork conditions and AssetAttacked conditions.

Model Strategy for Managing Attacks

The following "model" applies to educational scenarios in which the designer does not want to overwhelm the player with attacks of different types. These types of scenarios are most effective when the designer is able to focus the player on a series of issues, gaining understanding of each vulnerability before confronting the next.

Design the attack sequence logic as described above in the introduction to "Managing Attacks." The goal is to let the attack types progress based on the player's ability to defend against each attack type. In general, do not attempt to start and stop attacks as a means of giving the player warnings. Sometimes however it is effective to suspend attacks at the start of some phases.

Where motives are large (and compromise is fatal), set asset costs to zero. This keeps the attacks from appearing in the attack log and lets you use the attack events as conditions in warning triggers. Include the asset name in the appropriate Lose Trigger to generate an attack log entry at the end of the game.

After some amount of initial handholding, start the attacks with a predefined sequence and frequency.

Provide players warnings via tickers that include both the AssetAttacked condition and the AssetToNetwork condition (or some combination of other vulnerability assessment conditions). The former limits firing until this attack is occurring based on the designer's attack logic (e.g., don't start insider attacks until Internet attacks are dealt with), and the latter lets us withdraw the ticker if the player fixes the vulnerability. (As noted earlier, the game logic to withdraw tickers holds attack conditions as true, and thus any "and_not" conditions should be applied to AssetToNetwork conditions and not to AssetAttacked conditions.)

Condition Lose Triggers based on the occurrence of warnings (e.g., count intrinsic conditions) about potential catastrophic vulnerabilities.

Shotgun Advice

Limit player choices by giving them insufficient cash to brute force a solution.

Use the Loans trigger to let users continue even though they've spent their money, but don't let them take a big enough loan to brute force a solution.

Don't get bogged down in proving mechanical help, assume players have already played other scenarios that taught them the basics of the game.

Create a set of useful camera views for your scenario. Don't add too many views or it will take players too many steps to find what they need.

If two or more triggers are to go off under the same conditions, make all but one into a slave.

Don't try to create a lot of randomness unless you are creating a free-play scenario.

Don't mask attacks or use MaskAttack triggers – they are obsolete.

Things you can do to respond to an attack:

- Reduce player's cash (implicitly per asset or via explicit trigger)
- Light a computer on fire via a burn… trigger.
- Add a 3D model to the scene using the ShowObject trigger
- Have the user speak via the Speak trigger
- Load an encyclopedia page (changeEncyloTrigger) with a tutorial and use a message popup to tell the player to go there – or pop it updirectly.
- Play a movie via the PlayMovie trigger
- Use the DescribeAttack trigger to cause insiders to brag about what they did (does not work on outside attackers (yet)).
- Use the HoverComputer to display 2d images over computers.
- Use the SetRandom trigger to cause different response to attacks (see Random Values" above)

Things you can do to give player's a clue

- Ticker triggers (be sure they will erase if players correct problems)
- Use speaks triggers to have the characters describe problems
- Use computer diagnostic triggers to have computers describe their own problems
- Popup messages
- Load an encyclopedia page (changeEncyloTrigger) and use a message popup to tell the player to go there.
- Set user thoughts

Save games (via trigger) when moving to a new phase – but make sure all bonuses and loan adjustments are recorded BEFORE the game state is saved.

Set the Unhappiness Immunity for users to make them immune to gross unhappiness resulting from cameras, armed guards, etc. Otherwise, build in a lot of feedback to tell the player why the user is unhappy.

If users get up and wander when you don't expect them to, check their "wander" setting. If that is not set, then double check your workspace definitions. If you have more than one workspace defined close to the spot where the user's desk sits, the game engine may try to move the user to some other location.

Trigger Tables

AddSoftware	Add an application to a computer, e.g., to simulate a user constantly downloading malware.	
Parameter #	Name (Displayed in Tool	Description
TriggerText	Application	Name of the application to install.
SecondTriggerText	Computer	Name of the computer upon which to install the application.

AddZoneAccess	Grant a named user access to a zone. Intended for use where insiders get access even though player denies the access.	
Parameter #	Name (Displayed in Tool	Description
TriggerText	Zone	Name of the zone to which the user will have access.
SecondTriggerText	User	The user who is to have access.

AttackerName	Associate the given names with the attackers computer and the domain from which the attack occurs. These names are used when processing network filters.	
Parameter #	Name (Displayed in Tool	Description
TriggerText	ComputerName	Name of the attacker's computer.
SecondTriggerText	DomainName	Name of the domain from which the attack originates.

AttackTrigger	Cause a specific attack type to occur. If an asset is compromised, it is not attacked again as part of that trigger firing. However, other assets are continued to be attacked unless the motive is set to a prescribed value as described below.	
Parameter #	Name (Displayed in tool)	Description
TriggerText	AttackIdentifier	Descriptor for log entry.
Parameter	AttackType	Identifies a specific attack that is to occur. See Attack Types described below.
Parameter	AttackMotive	The motive of the attack. A value of "-2" will not override the motive associated with the assets. A value of "-3" is the same, but will also cease attacks on other assets from this firing after any attack succeeds. Also, a value of "-3" will keep this trigger from being considered for firing until the period denoted by its frequency is surpassed as measured subsequent to the most recent attack of any type. For "Bad Policy" attacks, -4 will limit malware attempts to Trojan horses and -5 will limit them to viruses. The former has no effect on availability.

BurnComputerAsset	Generate flames from the computer hosting the named asset.	
Parameter #	Name (Displayed in Tool	Description
TriggerText	Asset	The asset whose computer is to burn.
Parameter	Duration	0 puts the fire out 1 starts a fire and lets it burn >1 burns for given duration <0 burn until conditions of this trigger go false

BurnComputerUser	Generate flames from the computer assigned to a given user	
Parameter #	Name (Displayed in Tool	Description
TriggerText	User	The user whose computer is to burn.

Parameter	Duration	0 puts the fire out 1 starts a fire and lets it burn >1 burns for given duration <0 burn until conditions of this trigger go false

BurnHere	Generate flames at a given location	
Parameter #	Name (Displayed in Tool	Description
Parameter	X	X grid location
Parameter	Y	Height
Parameter	Z	Y grid location

CameraToUser	Move the camera to a user or staff	
Parameter #	Name (Displayed in Tool	Description
TriggerText	User	Name of the user or staff
Parameter	OneForFront	0 – to user's back 1 – to user's front
Parameter	Angle	Degree's to rotate clockwise
Parameter	Zoom	Zoom level (see viewpoints)
Parameter	Height	Camera height (see viewpoints)

CashTrigger	Adjust player's cash on hand	
Parameter #	Name (Displayed in Tool	Description
TriggerText	Text	Text to display in popup window (if any)
Parameter	Amount	Dollar value to adjust (+ or -)

CatalogueExists	Control whether item appears in the catalogue	
Parameter #	Name (Displayed in Tool	Description
TriggerText	Text	Name of catalogue item
Parameter	OneForExists	If "1", the item exists

ChangeAssetUsageTrigger	Adjust an asset goal "TargetUsage" of a given user.	
Parameter #	Name (Displayed in tool)	Description
TriggerText	User	Name of the user whose asset goal is to be adjusted.
SecondTriggerText	AssetGoal	The name of the asset goal whose TargetUsage is to be adjusted.
Parameter	TargetUsage	The new target usage to assign to the user's asset goal.

ChangeComponentName	Change the name of a component	
Parameter #	Name (Displayed in tool)	Description
TriggerText	oldName	Current name of component
SecondTriggerText	newName	The new name

ChangeCostMotive	Change the motive associated with a cost list entry	
Parameter #	Name (Displayed in tool)	Description
TriggerText	Asset	Name of the aset
Parameter	Entry	The entry number from the list
Parameter	Motive	The new motive

ChangeEnclyoTrigger	Sets the encyclopedia to show the page specified in the trigger the next time the player brings up the encyclopedia.	
Parameter #	Name (Displayed in tool)	Description
TriggerText	Target	URL of Specified page of encyclopedia.
Parameter	OneToLaunch	If 1, launch the encyclopedia.

ChangeLabelMotive	Change the attacker motive associated with a label	
Parameter #	Name (Displayed in tool)	Description
TriggerText	Label	The label whose attacker motive is to change
Parameter	Motive	The new value for the motive

ChangeUserDesc	Replace the description of a user	
Parameter #	Name (Displayed in tool)	Description
TriggerText	Name	Name of user whose description is being changed.
SecondTriggerText	NewDescription	New description for user.

ClearMalware	Remove all malware from all computers

ClearRegister	Reset a named register to its initial state. See the "Register"condition and the Question trigger.	
Parameter #	Name (Displayed in tool)	Description
TriggerText	Register	Name of the register to reset.

ClearTriggeredThoughts	Clear all triggered thoughts for a given user	
Parameter #	Name (Displayed in tool)	Description
TriggerText	Name	Name of user whose thoughts are to be cleared.

ComputerAddUser	Add a user account on the given computer. Intended for use on static computer whose name can't be changed by the player.	
Parameter #	Name (Displayed in tool)	Description
TriggerText	Computer	Name of the computer
SecondText	User	Name of user to add.

ComputerAvailability	Set the current availability for a computer	
Parameter #	Name (Displayed in tool)	Description
TriggerText	Name	Name of the computer
Parameter	Value	Availability expressed as a percentage.

ComputerDiagnostics	Finds the first computer with a diagnostic problem that matches a given mask, and uses speaks to describe the problem. Slaves to these triggers will only fire if a diagnostic is displayed.	
Parameter #	Name (Displayed in tool)	Description
TriggerText	Name	Not used yet
Parameter	Value	The mask that identifies the diagnostic conditions that would result in a speaks. See "attack properties".

Custom	Invokes a Java method called customJavaTrigger within the customClass class. The named register is assigned the returned unsigned byte value	
Parameter #	Name (Displayed in tool)	Description
TriggerText	Parameters	Space separated parameters to be passed to the java method as an array of strings.
Parameter	Register	The name of a register condition that is to be assigned the value returned from the java method.

DescribeAttack	Cause attacker to speak a description of an attack. For now, this only works for insider attacks.	
Parameter #	Name (Displayed in tool)	Description
TriggerText	Condition	Name of the condition that measures the attack. Note the condition might specify any attack rather than a specific attack type.
Parameter	OneToPause	If one, the game will pause while the user describes the attack. Otherwise, the value is the duration of the resulting speaks in seconds. A negative value moves the camera to the user.

EmailInstallRoot	Cause a named email client (or SSL Server as described below) to install the root of a subject CA. Intended for use to allow validation of some CA's certificate, e.g., to let a vendor validate a certificate issued by a CA bought by the player. Note that this trigger will only go off once per change in some PKI state, and its intended that they have a suitable frequency to install the root once the CA is put into place. Finally, note that slaves of this trigger type will not fire unless the CA is found and the root installation occurs.	
Parameter #	Name (Displayed in Tool	Description
TriggerText	Goal	Name of an email goal
SecondText	Computer	Where the root is installed
Parameter	OneForSSL	If "1", the computer is assumed to be an SSL server and the root will be installed for each SSL server application on that server.

HappyAdjTrigger	Add or subtract an amount to the player's base happiness	
Parameter #	Name (Displayed in tool)	Description
TriggerText	Text	The name of the user whose happiness is to be adjusted.

Parameter	Amount	Amount to alter the given user's happiness. Can be positive or negative.

HelpTipTrigger	Display a help tip balloon	
Parameter #	Name (Displayed in Tool	Description
TriggerText	Text	Text displayed for help.
Parameter	Xsize	Width of display balloon
Parameter	Ysize	Height of display balloon
Parameter	X	X of start
Parameter	Y	Y of start

HideSite	Hide an offsite office from the player and all of its components.	
Parameter #	Name (Displayed in tool)	Description
TriggerText	Zone	Name of the zone corresponding to the site to hide.
Parameter	OneToHide	1 = hide the site 2 = show the site

HoverComputerAsset	Cause a given image file to display when the mouse hovers over a computer containing the given asset.	
Parameter #	Name (Displayed in tool)	Description
TriggerText	Asset	Name of the asset whose associated computer is the target.
SecondTriggerText	imageFile	Name of file containing image. Should be a tga.
Parameter	Immediate	If non-zero, image displays for given duration (seconds) when the trigger goes off.

HoverComputerUser	Cause a given image file to display when the mouse hovers over a computer assigned to a given user.	
Parameter #	Name (Displayed in tool)	Description
TriggerText	User	Name of the user whose assigned computer is the target.
SecondTriggerText	imageFile	Name of file containing image. Should be a tga.
Parameter	Immediate	If non-zero, image displays for given duration (seconds) when the trigger goes off.

InstallMalware	Unconditionally install malware (randomly selected) on a specified computer	
Parameter #	Name (Displayed in tool)	Description
TriggerText	Name	Name of the computer
Parameter	malwareType	0 = visible to scan & net 1 = invisible to net 2 = invisible to scan 3 = invisible to both

InternetDown	Cause the Internet to become unusable for communication	
Parameter #	Name (Displayed in tool)	Description
Parameter	OneForDown	1 = Internet is down 0 = Internet is back up

KeyExposed	Simulate exposure of private PKI key in a manner that is immediately discovered. Use of a CRL mitigates resultant attacks. Currently only supported for email. Attacker still must be able to compromise the email asset (but is not deterred by signing or encryption.)	
Parameter #	Name (Displayed in Tool	Description
TriggerText	User	User whose email goal is the subject of the exposed certificate key
SecondTriggerText	Goal	User goal
ThirdTriggerText	Software Type	Currently, only "email client" is supported.

LabManual	Set the name of the file to display when "E" is pressed. Intended for use in displaying lab manuals. Overrides value set in Scenario form.	
Parameter #	Name (Displayed in tool)	Description
TriggerText	Link	File name of manual

Loan	If the global NoLoans is set, and a player tries to buy equipment beyond his cash on hand, then if the purchase price is less than the value set by this trigger, the player will be given an option of taking out a loan that would leave this value as the player's cash on hand, less the cost of the equipment. Players are only able to take loans to replenish losses due to attacks, cash triggers and goal penalties. Thus, if the designer provides enough money for a limited set of equipment, and the player loses money, the player can still purchase the limited equipment via a loan – but the player cannot take loans beyond the intended capitalization.	
Parameter #	Name (Displayed in tool)	Description
Parameter	Amount	Amount the player would be left with, less the cost of the item.

LogTrigger	Generate a log entry containing specific text.	
Parameter #	Name (Displayed in tool)	Description
TriggerText	Text	Text to include in the log entry.

LoseTrigger	Results in the player losing the game and the given being displayed. If no text is given, the most recent attack log entry is displayed.	
Parameter #	Name (Displayed in Tool	Description
TriggerText	Text	Text displayed upon losing.
SecondTrigger Text	OptAssetName	Optional name of an asset intended for use with zero-cost asset attacks to cause creation of an AttackLog entry (which had not been made for attacks due to the zero cost attribute of the asset.)

MaskAttackTrigger	Use to turn masks on or off for all the different attack types. NOTE: this and attack masks will be phased out. It is intended that all attacks be via AssetAttack triggers.	
Parameter	Value	See "Attack Types"

Parameter	ValueMask	Set mask to true or false. 1= mask this attack type 0 = don't mask this attack

MessageTrigger	Results in a pop-up message being displayed.	
Parameter #	Name (Displayed in tool)	Description
TriggerText	Text	The text displayed to the user when the trigger occurs.

Move User	Cause a named user to relocate to a different workspace	
Parameter #	Name (Displayed in tool)	Description
TriggerText	Text	Name of the user to move.
Parameter	Position	New position index
Parameter	OneNoTransport	If the value is "1", then the character is not transported to the new workspace.

ObjectTexture	Change the texture of one of the dynamic objects that are loadable via the "ShowObject" trigger. One ObjectTexture trigger must fire prior to the first ShowObject trigger for any given object. Subsequent triggers may change an object's texture, which takes effect on the next "ShowObject" trigger.	
Parameter #	Name (Displayed in tool)	Description
TriggerText	Texture	Name of a tga file (without extension) that contains the new texture.
Parameter	Shape	The shape whose texture is to change. See the "ShowObject" trigger

PacketXform	Append a given network packet sample to the packet log associated with a given component after transforming component names as defined in a PacketXform form.	
Parameter #	Name (Displayed in Tool	Description

TriggerText	xformName	Name of the form that defines the name transformations (see the Xform form definition). Note the first entry of this form names the component whose packet log is to be extended.
SecondTriggerText	sampleName	The name of the network packet sample.

PauseGame	Pause the game	
No parameters		

PlayMovie	Play movie without player intervention	
Parameter #	Name (Displayed in tool)	Description
TriggerText	file	Name of the movie file in the game/exe directory.

ProdAdjTrigger	Add or subtract an amount to the player's base productivity	
Parameter #	Name (Displayed in tool)	Description
TriggerText	Text	The name of the user whose productivity is to be adjusted.
Parameter	Amount	Amount to alter the given user's productivity. Can be positive or negative.

Question	Ask the player a question using a popup. Embedding strings: (a), (b), and (c) will create a multiple choice layout. Be sure to leave spaces around those strings. (See the LiveWithMacros scenario for an example). Otherwise, the question will appear with "yes" and "no" buttons. Scenario designers must assess the register value given as the condition, and provide players with suitable responses (e.g., via user speech.)	
Parameter #	Name (Displayed in tool)	Description

TriggerText	Text	The text of the question and answers.
SecondTriggerText	ConditionName	Name of the register condition to modify based on player response. See the “register” condition.

QuestionMult	Cause a named Question to be displayed (see the question form). These triggers are intended to simplify management of simple multiple choice questions and the game response to player answers.	
Parameter #	Name (Displayed in tool)	Description
TriggerText	Name	The name of the Question that is to be displayed.
SecondTriggerText	ConditionName	Name of the register condition to modify based on player response. See the “register” condition.
ThirdTriggerText	User Name	Optional user name. If provided, responses to player answers will be spoken by the named user. Otherwise, response are displayed as messages.
Parameter	OneToFreeze	Only used if a user is named above. See Speak Trigger
Parameter	Camera Index	Only used if a user is named above. See Speak Trigger

QuestionUser	QuestionUser triggers tie named Question triggers to users such that players can retry the question by right clicking on the user. The QuestionUser trigger defines a set of “good” answers that will cause the link between the user and the Question trigger to be broken. As long as the link is in place, the user is prevented from achieving ANY goals, and right clicking on the user will repose the question.	
TriggerText	User Name	The name of the user to link to the Question trigger
SecondTriggerText	Question	The name of the Question trigger that is to be linked to the user.
Parameter	GoodAnswer1	First “good” answer.
Parameter	GoodAnswer2	Second “good” answer.

Parameter	GoodAnswer3	Third "good" answer.

QuitGameTrigger	Game immediately shuts itself down.	

RemoveAsset	Remove an asset from wherever it might be. The asset will entirely disappear from the scenario. NOTE: the game will crash if there any goals that require this asset. NOTE further: the game will crash if an asset usage trigger causes a goal to access this asset.	
TriggerText	Asset Name	The name of asset remove.

ResetAssetAttacked	Reset an asset attacked condition, intended to be done after a trigger goes off on the condition so that other triggers do not go off unless the asset is attacked again. Note that a trigger that goes off due to an asset being attacked will not go off again unless there is a new attack – regardless of whether the ResetAssetAttacked trigger is used. BEWARE, this trigger may prevent some other trigger from firing because it will keep the second trigger from seeing that the asset was compromised (trigger assessment is an unpredictable sequence). Similarly, it may cause some other trigger to fire despite having a condition that prevents firing if the asset is attacked. Use this trigger with care.	
TriggerText	Condition Name	The name of the AssetAttacked condition to reset.

ResetEncyclopedia	Reset the encyclopedia override that was set with the changeEncyloTrigger.

ResetUser	Move a user back to its starting position	
Parameter #	Name (Displayed in Tool	Description
TriggerText	User	Name of the user

SaveGameTrigger	Save current game state to named file.	
TriggerText	Fid	Name of the file, must include ".sdf" extension. If the name begins with %sdfId (case sensitive), that string will be replaced with the name of the Scenario. This is intended to distinguish saved games for different scenarios in the same project that share save game triggers.

SetCameraToIndex	Set the camera to the given index. See "Generating Camera Viewpoints"	
Parameter	Index	The viewpoint index (starting with zero) that the camera is to be set to.

SetObjectiveStatus	Set an objective as being achieved or not	
Parameter #	Name (Displayed in Tool	Description
TriggerText	Objective	Name of the objective
SecondTriggerText	Message	Optional message to display
Parameter	OneForMet	1 = met; 0 = not met

SetPhase	Change the current phase of the game	
Parameter #	Name (Displayed in Tool	Description
TriggerText	NewPhase	Name of new phase
SecondTriggerText	Message	Optional text to display

SetRandom	Change the value of a RandomValue condition. Note that the values are not strictly random because values will not repeat until the full set of random values is cycled through. For example, if the range is 0 to 5, each of the six integers will be utilized before a new random sequence is generated.	
Parameter #	Name (Displayed in Tool	Description
TriggerText	Condition Name	Name of the Random Value condition.

SetRegister	Sets a named register to a non-zero value. Intended to be used to skip selected question triggers	
Parameter #	Name (Displayed in Tool	Description
TriggerText	Condition Name	Name of the register to set

SetUserThought	Set the thought for a user. These user thoughts will be erased when the conditions of this trigger go false.	
Parameter #	Name (Displayed in Tool	Description
TriggerText	User	Name of user
SecondTriggerText	Thought	The user thought
Parameter	Priority	The priority of the thought. The higher the value, the higher the priority. See the list of User Thought Priorities

SetZoneBlocked	Prevent a user (animated character) from entering a zone. Intended for use with checkpoints.	
Parameter #	Name (Displayed in Tool	Description
TriggerText	Zone	Name of the zone
SecondTriggerText	User	The user to block. Blank means all users
Parameter	Value	0 = unblock 1 = block all users 2 = block just the named user

ShowObject	Show a triggered object (named by the shape). Currently, these are all billboards. If the x value is negative, the object is hidden. Otherwise, if the object is already showing, its texture may change based on the most recent ObjectTexture trigger for the given shape.	
Parameter #	Name (Displayed in Tool	Description
Parameter	Shape	Integer, see table
Parameter	X	Grid x position
Parameter	Y	Grid y position
Parameter	Angle	0 N 1 E 2 S 3 W
Parameter	Height adjust	

Parameter	X adjust	trial & error
Parameter	Y adjust	

SmokeUser	Cause smoke to rise off of user's head	
Parameter #	Name (Displayed in Tool	Description
TriggerText	User	The user whose head is to generate smoke
Parameter	Duration	0 stops the smoke 1 starts a smoke and lets it billow >1 smokes for given duration <0 will smoke until conditions of this trigger go false.

SpeakTrigger	Like help tips, but balloons come from users.	
Parameter #	Name (Displayed in Tool	Description
TriggerText	User	Name of user
SecondTriggerText	Words	What the user says
Parameter	OneToFreeze	Set to 1 to pause the game. The camera will go to the user and the speak will display until player unpauses the game, or clicks the speak. Other positive values are the duration the balloon will stay up. A negative value moves the camera to the user.
Parameter	Camera Index	Move camera to this camera index. If this value is set, and OnetoFreeze is negative, then the named user is also selected in the info panel. Ignored if OneToFreeze is one.
Parameter	Erase	If one, the speak will be erased if the conditions that caused it evaluate to false.

TickerTrigger	Results in a ticker message being queued for display.	
Parameter #	Name (Displayed in tool)	Description
TriggerText	Text	The text displayed to the user when the trigger occurs.
Parameter	Repeat	The number of times the message is to cycle through the ticker.
Parameter	Erase	Erase ticker if conditions change such that trigger would not now go off (not useful with AssetAttacked or CheckPointFail or TriggerGoneOff conditions.)

TrustAdjTrigger	Alter a users base trustworthiness	
Parameter #	Name (Displayed in Tool	Description
TriggerText	User	Name of user
Parameter	TrustChange	New base trustworthiness

UserStay	Cause a user or staff to stay at terminal point of latest walk.	
Parameter #	Name (Displayed in Tool	Description
TriggerText	User	Name of user or staff
Parameter	1 0	Character will stay Character free to wander

UserWhineTrigger	Display message if user is unable to achieve goals as follows: 0 – achieve goal entirely assigned computer 1 – achieve goal using network 2 – must use other computer in same zone 3 – must leave zone 4 – no access	
Parameter #	Name (Displayed in Tool	Description
TriggerText	User	The user whose goal achievement is assessed.
Parameter	zto4	Least access per above
Parameter	zto4	Most access per above

ViewpointTab	Enable or disable tabbing to a viewpoint.	
Parameter #	Name (Displayed in Tool	Description
Parameter	Index	Which viewpoint to enable or disable.
Parameter	OneToSkip	0 – Viewpoint can be tabbed to; 1 – tabbing to the viewpoint is disabled.

VPNCA	Cause an indirectly named CA to issue a certificate for a named VPN mechanism. The CA is named in terms of a user's access to a given asset, i.e., "whatever CA facilitates the data side of Joe's access to the Roadmap." Intended for use with VPN mechanisms controlled by rogue users within an enterprise. ***NOTE*** the VPN mechanism is named by the name of this trigger. Note also that this trigger will only go off once per change in some VPN state, and its intended that they have a suitable frequency to issue the cert once a CA is put into place.	
Parameter #	Name (Displayed in Tool	Description
TriggerText	User	The user whose goal helps to name the CA
SecondTriggerText	Asset	The asset accessed by the user as part of the goal that identifies the CA that is to issue the certificate.
Parameter	OneForDataCA	If the value is one, then the CA is the same as that used by the data-side VPN that facilitates the goal. Otherwise, it is the user-side VPN mechanism.
Parameter	OneForGW	If one, the named VPN mechanism is a gateway.

XCertCA	Cause a given CA to cross certify all of the enterprise's CAs. Intended for use to cross certify some CA's certificate, e.g., to cause a remote vendor's site to sign the cert of the player's CA. *** Note also that this trigger will only go off once per change in some VPN state, and its intended that they have a suitable frequency to issue the cert once a CA is put into place. Finally ?, note that slaves of this trigger type will not fire unless the enterprise CA(s) are found and the cross certification occurs.	
Parameter #	Name (Displayed in Tool	Description
TriggerText	User	The user whose goal helps to name the CA
SecondTriggerText	Asset	The asset accessed by the user as part of the goal that identifies the CA that is to issue the certificate.
Parameter	OneForDataCA	If the value is one, then the CA is the same as that used by the data-side VPN that facilitates the goal. Otherwise, it is the user-side VPN mechanism.

VPNxCertCA	Cause an indirectly named CA to have its CA certificate signed by the CA whose name matches the name given to the trigger. The subject CA is named in terms of a user's access to a given asset, e.g., "whatever CA facilitates the data side of Joe's access to the Roadmap." Intended for use to cross certify some CA's certificate, e.g., to cause a remote vendor's site to sign the cert of the player's CA. ***Note the name of the signing CA is the name given to the trigger. Note also that this trigger will only go off once per change in some VPN state, and its intended that they have a suitable frequency to issue the cert once a CA is put into place. Finally ?, note that slaves of this trigger type will not fire unless the named CAs are found and the cross certification occurs.	
Parameter #	Name (Displayed in Tool	Description
TriggerText	User	The user whose goal helps to name the CA

SecondTriggerText	Asset	The asset accessed by the user as part of the goal that identifies the CA that is to issue the certificate.
Parameter	OneForDataCA	If the value is one, then the CA is the same as that used by the data-side VPN that facilitates the goal. Otherwise, it is the user-side VPN mechanism.

VPNLimit	Limit the number of VPN mechanism that can be deployed at a site (or offsite). Intended to force players to use Certificate Policies.	
Parameter #	Name (Displayed in Tool	Description
TriggerText	Zone	The name of the zone of the entire office – or the offsite that is to have limited numbers of VPN mechanisms.
Parameter	NumVPN	The maximum number of deployable VPN mechanisms

WalkComputerUser	Cause a user or staff to walk to a computer assigned to a given user	
Parameter #	Name (Displayed in Tool	Description
TriggerText	User	Name of user or staff who is to walk
SecondTriggerText	User's Computer	Name of user whose computer is to be visited.

WalkComputerAsset	Cause a user or staff to walk to a computer containing a given asset	
Parameter #	Name (Displayed in Tool	Description
TriggerText	User	Name of user or staff who is to walk
SecondTriggerText	Asset's Computer	Name of asset whose associated computer is to be visited.

WalkToUser	Cause a user or staff to walk to another user	
Parameter #	Name (Displayed in Tool	Description
TriggerText	User	Name of user or staff who is to walk
SecondTriggerText	DestUser	Name of the user who is to be visited.

WinTrigger	Results in the player winning the game and the text from the "DebriefWin" being displayed.	
Parameter #	Name (Displayed in Tool	Description
TriggerText	Text	Text displayed upon winning.

Condition Tables

AllAssetGoalsMet	All user asset goals are met.

AssetAttackCount	Whether an asset (named via an AssetAttacked condition) has been attacked a given number of times. The count is a trigger condition parameter and is evaluated with a "greater than" operator.	
Parameter	Name (Displayed in tool)	Description
ConditionText	Condition	The name of an AssetAttacked condition that names the asset
Parameter	Count	The exact number of times the asset was attacked for the condition to be true (unless trigger condition parameters are used)

AssetAttacked	Whether a specific asset has been attacked, optionally using a given attack type and attack property. For any given trigger, this condition will be true only once per attack, e.g., a trigger will not fire twice due to the same instance of an attack. The attack property is a trigger condition parameter value for this condition. Note that the condition's attack property must be left blank for the trigger condition parameter to work.	
ConditionText	AssetName	Name of asset attacked. A blank means any asset, but AttackType must then be -1 (see below).
Parameter	AttackType	See "Attack Types". A value of -1 denotes any attack type.
Parameter	MinMotiveValue	Minimum motive for attack.
Parameter	MaxMotiveValue	Maximum motive for attack.
Parameter	AttackProperty	Blank means any property. Otherwise treated as a mask that is ANDed with an ORing of each property of the attack (e.g., sniffed password.) NOTE this is not intended to be used – place attack property condition in triggers as condition parameters.
Parameter	WithinSeconds	If set, once true, the condition will be held true for this number of game time seconds.

AssetComputerHasMAC	Whether the computer containing the named asset enforces a MAC policy.	
Parameter	Name (Displayed in tool)	Description
ConditionText	Asset	Asset name

AssetComputerHasPolicy	Whether the computer containing a given asset has a given policy.	
Parameter	Name (Displayed in tool)	Description
ConditionText	Asset	Asset name
SecondConditionText	PolicyOfConfiguration	See AssignedComputerHas

AssetComputerSLNet	Whether the computer containing a given asset has a network connection to the given user's computer with the given label.	
Parameter	Name (Displayed in tool)	Description
ConditionText	Asset	Asset name
SecondConditionText	User	User whose computer is connected to the asset's computer
ThirdConditionText	Label	The label to test

AssetComputerVPNNeedsMeasuredBoot	Whether the computer containing a given asset is configured with a VPN client that relies on a measured boot.	
Parameter	Name (Displayed in tool)	Description
ConditionText	Asset	Asset name

AssetInference	Given a high asset and a low asset, does a join on a given common column yield any columns from the low asset that mostly have the same value and a matching column name in the high table – thereby suggesting an inference?	
Parameter	Name (Displayed in tool)	Description
ConditionText	High Asset	Name of the high asset
SecondConditionText	Low Asset	Name of the low asset
ThirdConditionText	Common column	Name of the common column.

AssetInZone	Whether a given asset is in a given zone.	
Parameter	Name (Displayed in tool)	Description
ConditionText	Asset	Asset name
SecondConditionText	Zone	Zone name (blank implies any zone)

AssetMissingGroup	Determine if an asset's ACL names a group that is not know the computer hosting the asset.	
Parameter	Name (Displayed in tool)	Description
ConditionText	Asset	Asset name

SecondConditionText	Group	The group to be tested. A blank is interpreted as "any group".

AssetMissingUser	Determine if an asset's ACL names a user that is not know the computer hosting the asset.	
Parameter	Name (Displayed in tool)	Description
ConditionText	Asset	Asset name
SecondConditionText	User	The user to be tested. A blank is interpreted as "any user".

AssetsOnSameComputer	Are two assets on the same computer?	
Parameter	Name (Displayed in tool)	Description
ConditionText	Asset1	Asset name
SecondConditionText	Asset2	Name of the second asset

AssetSuffersDoS	Is the asset currently suffering from a successful denial of service attack,?	
Parameter	Name (Displayed in tool)	Description
ConditionText	Asset	Asset name

AssetToInternetNoVPN	Whether a given asset travels the internet as part of some user's goal without going through a vpn gateway. The dynamically evaluated answer to this condition is a greater than comparison and the value is the assurance of the protection of the asset on the wire (e.g., based on vpn encryption). If a dynamic condition parameter is used with this condition in a trigger condition list, than it usually makes sense to also have another identical condition in the list to ensure the asset is protected by a VPN in the first place.	
Parameter	Name (Displayed in tool)	Description
ConditionText	Asset	Asset name
Parameter	Mode	1 – read 2 -- modify

AssetToNetwork	Whether a named asset can be reached via a given network for a given mode based on the asset's motives. The AttackProperty is the trigger condition parameter.	
Parameter	Name (Displayed in tool)	Description
ConditionText	Asset	Asset name
SecondConditionText	Network	Network name
Parameter	mode	1 = read 2 = write
Parameter	AttackProperty	Blank means any property. Otherwise treated as a mask that is ANDed with an ORing of each property of the attack (e.g., sniffed password.)

AssetToNetworkByFilterType	Whether a named asset can be reached via a given network through a software filter. The AssetToNetworkByFilterType has two intended uses: 1) warning players prior to triggering attacks; 2) determine if player has blocked a port that is needed to achieve a goal. Note that when used to determine if attacks could occur – other counter measures may force this condition to false – even if no filters block the given software type. If goal is true, then other countermeasures (e.g,. ACL limits) are ignored and the test is whether the software type reaches the computer.	
Parameter	Name (Displayed in tool)	Description
ConditionText	Asset	Asset name
SecondConditionText	Network	Network name
ThirdConditionText	SoftwareType	Software Type, none means any
Parameter	Direction	Only used when goal is set to one. Otherwise the direction is from the network. 0 = from network, 1 = to network, 2= both

Parameter	Goal	Set to one if the condition is to determine if filters are blocking an asset goal. Otherwise leave blank.

AssetToNetworkFilterCount	The minimum number of filtered software types via which the named asset can be reached via a given network. The count value is a trigger condition parameter.	
Parameter	Name (Displayed in tool)	Description
ConditionText	Asset	Asset name
SecondConditionText	Network	Network name
Parameter	Mode	0 = from, 1 = to, 2= both
Parameter	Count[1]	Minimum quantity of filtered software types.

AssetVulnerable	Whether a named asset would be compromised by an attack of the given type The AttackProperty is the trigger condition parameter.	
Parameter	Name (Displayed in tool)	Description
ConditionText	Asset	Asset name
Parameter	AttackProperty	Blank means any property. Otherwise treated as a mask that is ANDed with an ORing of each property of the attack (e.g., sniffed password.)

AssetWsHasIdDevice	Whether the workstation containing a named asset has an associated ID Device, and if so, its type. Trigger condition parameters assess to the type.	
Parameter	Name (Displayed in tool)	Description
ConditionText	Asset	Asset name
Parameter	Type	The ID device type (see ZoneHasEquip for values)

AssetZoneHasGuard	Whether the zone containing a given asset has a guard	
Parameter	Name (Displayed in tool)	Description
ConditionText	Asset	Asset name
Parameter	OneForDoor	0 – patrolling guard 1 – door guard

AssetZoneHasPolicy	Whether the zone containing a given asset has a specific policy in place.	
Parameter	Name (Displayed in tool)	Description
ConditionText	Asset	Asset name
SecondConditionText	PolicyOfZone	PermitEscortedVisitors: Badges: ModerateIrisScanner: ExpensiveIrisScanner: XRayPackages: SurveillanceCameras: Re-enforcedWalls: ModeratePerimeterAlarms: ExpensivePerimeterAlarms: ProhibitPhoneDevices: ProhibitMedia: CipherLockOnDoor: KeyLockOnDoor: VisualPeopleInspection: PatrollingGuard: GuardAtDoor: Receptionist

AssetZoneHasSecurity	Whether the zone containing the named asset has at least a given physical security	
Parameter	Name (Displayed in tool)	Description
ConditionText	Asset	Asset name
Parameter	Value	Minimum security strength

AssetZoneUserAccess	Whether a given user can enter the zone containing a named asset.	
Parameter	Name (Displayed in tool)	Description
ConditionText	Asset	Asset name
SecondConditionText	User	Name of user whose access is being tested.

AssetZoneClearanceAccess	Whether a user with a given clearance can enter the zone containing a named asset.	
Parameter	Name (Displayed in tool)	Description
ConditionText	Asset	Asset name
SecondConditionText	Clearance	Label associated with clearance being tested.

AssignedComputerHas	Tests whether a specified user's assigned computer has a specific policy or configuration setting.	
ConditionText	UserName	Name of a user, can be "*".

SecondConditionText	PolicyOfConfiguration “:”	The name of the computer setting to test against. Must use internal game name. ****Procedural Settings***** ProtectWithACL: WriteDownPassword: LockorLogoff: PasswordLength:Long PasswordLength:Medium PasswordLength:Short PasswordLength:None PasswordCharacterSet:Any PasswordCharacterSet:Moderate PasswordCharacterSet:Complex PasswordChangeFrequency:two PasswordChangeFrequency:six PasswordChangeFrequency:twelve PasswordChangeFrequency:never NoEmailAttachmentExecute: NoExternalSoftware: NoUseOfModems: NoMediaLeaveZone: NoWebMail: ApplyPatches: LeaveMachinesOn: NoPhysicalModifications: UserBackup: HoldsUserAsset: UpdateAntivirus:None UpdateAntivirus:Regular UpdateAntivirus:Automatic ***Configuration Settings**** OffsiteBackup: AdminBackup: RemoteAuthentication: LocalAuthentication: EnforcePasswordPolicy: MinimalLogging: DetailedLogging: UseOnTimePasswordToken: UseBiometrics: UseSmartcard: ScanEmailAttachments: StripEmailAttachments: AutomaticLockLogout: SelfAdminister: S SelfAdministerMAC:

		SelfAdministerMAC: UserRunsPrivileged: BlockRemovableMedia: BlockLocalStorage: UpdatePatches:None UpdatePatches:Regular UpdatePatches:Automatic UpdatePatches:AsReleased ManagedAntivirus: CM:Weak CM:Moderate CM:Strong MeasuredBoot: BrowserSettings:Loose BrowserSettings:Normal BrowserSettings:Strict EmailSettings:Loose EmailSettings:Normal EmailSettings:Strict

AssignedComputerZoneHasPolicy	Determine whether the zone containing a given user's assigned computer has a given policy	
Parameter	Name (Displayed in tool)	Description
ConditionText	User	User name
SecondConditionText	PolicyOfZone	See AssetZoneHasPolicy

AvgUserHappiness	The current average user happiness of all workers, not including IT and security, has a given relationship to a given value over a given period of time	
Parameter (id)	Name (Displayed in tool)	Description
Parameter	relation0forLessThan	0 means less than test, 1 means greater than test.
Parameter	TestValue	The amount of happiness to test against.

AvgUserProd	The current average user productivity of all workers, not including IT and security, has a given relationship to a given value over a given period of time	
Parameter (id)	Name (Displayed in tool)	Description
Parameter	relation0forLessThan	0 means less than test, 1 means greater than test.

Parameter	TestValue	The amount of productivity to test against.

AvgCash	Enumeration value that defines the condition class of this condition. Average is taken over a period of 14 days.	
Parameter	relation0forLessThan	0 means less than test, 1 means greater than test.
Parameter	TestValue	The amount of cash to test against.

CheckPointFail	Whether a given user has failed to pass a checkpoint to a given zone. This condition is primarily intended to be used with conditional trigger parameters (i.e., where the value of "mode" below is provided as a parameter within the trigger itself.)[7] After a given trigger has been assessed (i.e., is ready to go off should the conditions warrant), any CheckPointFail conditions that were non-zero will subsequently assess as zero (but only for that given trigger) until the user is denied or permitted access to the zone again.	
Parameter	Name (Displayed in tool)	Description
ConditionText	Zone	Zone name
SecondConditionText	User	User name, blank means any user.

[7] A notable exception is when the condition is to be referenced in a EventSequence condition.

Parameter	Mode	Access Failure Mode: -2 allowed in 0 not tested, or reset 1 not on access list 2 visitor, but no escort 3 zone policy does not match equipment 4 ZoneBlocked trigger set to one (block all users) 5 ZoneBlock trigger set to two and zone blocked for this user 6 Zone user fails all its asset goals having the "visitor" attribute (intended to group goals such that only one must be met.) 7 User is not a visitor, zone has some ID equipment and zone user fails all asset goals.

CompanyHasComputers	Whether the quantity of computers in the enterprise falls between two values, inclusive.	
Parameter (id)	Name (Displayed in tool)	Description
Parameter	Min	Minimum number
Parameter	Max	Maximum number

CompanyHasDevices	Whether the quantity of devices in the enterprise falls between two values, inclusive.	
Parameter (id)	Name (Displayed in tool)	Description
Parameter	Min	Minimum number
Parameter	Max	Maximum number

CompanyHasMACComputers	Whether the quantity of MAC enforcing computers in the enterprise falls between two values, inclusive.	
Parameter (id)	Name (Displayed in tool)	Description
Parameter	Min	Minimum number
Parameter	Max	Minimum number

CompanyHasVNPs	Whether the quantity of VPN gateways in the enterprise falls between two values, inclusive. When assessed against trigger condition parameters, a "greater than" operation is assessed.	
Parameter (id)	Name (Displayed in tool)	Description
Parameter	Min	Minimum number
Parameter	Max	Minimum number

CompanyHasWorkstations	Whether the quantity of workstations in the entire game falls between two values, inclusive. Does not count hidden computers.	
Parameter (id)	Name (Displayed in tool)	Description
Parameter	Min	Minimum number
Parameter	Max	Maximum number

CompareTriggerCounts	Whether one trigger has gone off more times than another	
Parameter (id)	Name (Displayed in tool)	Description
Parameter	First trigger	Trigger name
Parameter	Second trigger	If the first trigger has gone off more times than this trigger, then condition is true.

ComputerDiagnosticUser	The attack profile (if any) of the computer assigned to a given user. Only intended for use with trigger condition parameters.	
Parameter (id)	Name (Displayed in tool)	Description
ConditionText	User	Name of user whose computer is to be tested.

ComputerHasVPNClient	Whether the computer assigned to the given user has a VPN client, i.e., the "vulnerable network" is set. Note use of connection profiles can bypass VPN client for a goal. Also see "GoalUsedVPN".	
Parameter (id)	Name (Displayed in tool)	Description
ConditionText	User	Name of user whose computer is to be tested.

ComputerOff	Whether the given user's computer is turned off.	
Parameter (id)	Name (Displayed in tool)	Description
ConditionText	User	Name of user whose computer is to be tested.

ComputersAreConnected	Whether the computer containing an asset is connected to a given user's assigned computer without considering filters, VPNs, etc.	
Parameter (id)	Name (Displayed in tool)	Description
ConditionText	Asset	Name of the asset contained within the first computer.
SecondConditionText	User	Name of the user assigned to the second computer.

CrossCert	Whether a given CA (or any dominate CA in the same domain) has signed a subject CA's certificate (or any of the subject's dominate CA's that are within the same domain.)	
Parameter (id)	Name (Displayed in tool)	Description
ConditionText	Goal	Name of the asset goal whose achievement relies on the given CA. This is how the first CA is named.
SecondConditionText	User	User whose goal is met.
ThirdConditionText	Subject CA	The name of the subject CA

Custom	Whether a specific custom condition has been met. See "Custom Condition" below. Intended only for tutorial mode of the game.	
Parameter (id)	Name (Displayed in tool)	Description
Parameter	Condition #	The condition number to test as described below in "Custom Condition"

DomainHasCA	Whether a given domain has a CA	
Parameter (id)	Name (Displayed in tool)	Description
ConditionText	Domain name	The name of the domain to be tested.

Parameter	OneForTopLevel	If 1, then the test is against top level domains, e.g., here.myco would match there.myco.

EmailProtected	Whether a given email asset is protected with a given protection type (encrypted or authenticated).	
Parameter (id)	Name (Displayed in tool)	Description
ConditionText	Asset	Name of email asset whose protection is to be measured
SecondConditionText	Attacker	Name of the attacker, blank means external attacker.
Parameter	Mode	1=encrypted 2=authenticated 3=both
Parameter	Motive	Attacker motive to compromise the asset. If missing or negative and mode is write, then the condition is true if the sender signs the email. If missing or negative, asset motive is used. Otherwise, only for measuring strength of crypto algorithm (not yet used).
Parameter	Smart Card	If positive, the condition is only true if the protection is provided via a smartcard.

EventSequence	Whether one event has occurred since the most recent occurrence of some other event.	
Parameter (id)	Name (Displayed in tool)	Description
ConditionText	First condition	Name of first condition

SecondConditionText	Second condition	Name of second condition. If the event represented by this condition occurred subsequent to the most recent occurrence of the event named by the first condition, then the EventSequence condition is true. Game time is used, and thus events that occur while paused can't be accurately measured. If the times are equal, then the condition is assessed as true. And the condition will not be true unless the First Condition has occurred at least once.

FilterNamesComponent	Whether some filter somewhere names the given component as an exception	
ConditionText	Component	Name of the component

GameOnScreen	Whether a given screen is being displayed.	
Parameter (id)	Name (Displayed in tool)	Description
Parameter	Screen[1]	DEBRIEF 1 OFFICE 2 NETWORK 3 COMPONENT 4 ZONE 5 USER 6 ASSET 7 GAME 8 BUY 9 SOFTWARE 10 ITMANAGE 11 OBJECTIVE 12 FILTER 13 ACCESS LIST 14 LABEL SCREEN 15 LINK Encryptor 16 VPN config 17 Email config 18 SSL Client 19 SSL Server 20 SL Network 21 ML Network 22 ATTACK LOG 23 Cyber Chark 24

GameStateInfo	Whether the game is paused	
Parameter (id)	Name (Displayed in tool)	Description
Parameter	OneForPaused	1=paused 0=running

GoalHasRoot	Whether the workstation used to achieve a given user's goal has a root certificate from the given domain	
Parameter (id)	Name (Displayed in tool)	Description
Condition Text	Goal	Name of the goal to be tested
Condition Text	User	User name whose goal is to be tested
Condition Text	Domain	The domain against which to test the root certificates.

GoalUsedVPN	Whether a given user's goal used a VPN	
Parameter (id)	Name (Displayed in tool)	Description
Condition Text	User	User name whose goal is to be tested
Condition Text	Goal	Name of the goal to be tested

GoalUsesSSL	Whether a given user's goal uses SSL	
Parameter (id)	Name (Displayed in tool)	Description
Condition Text	Goal	Name of the goal to be tested
Condition Text	User	User name whose goal is to be tested

GoalSW	Whether a given goal is being met using a specified application. This is only meaningful for shared goals requiring compatible applications.	
Parameter (id)	Name (Displayed in tool)	Description
Condition Text	Goal	Identifies the goal
Condition Text	Application	Identifies the appliation

ItSecStatus	The percentage staffed of IT staff	
Parameter (id)	Name (Displayed in tool)	Description

Parameter	1forsec	0 = Tech support 1 = security
Parameter	Min	The minimum staffing for the condition to be true.

MaxCashOnHand	Cash test. True if the player's cash is at least the tested value.	
Parameter (id)	Name (Displayed in tool)	Description
Parameter	MaxCashValue	The amount of cash to test against.

MinCashOnHand	Cash test. True if the player's cash is less than the tested value.	
Parameter (id)	Name (Displayed in tool)	Description
Parameter	TestValue	The amount of cash to test against.

NetworkKeyLife	Determine the age of the oldest link encryptor key used to protect data on a given network.	
Parameter	Name (Displayed in tool)	Description
ConditionText	Network	Network name
Parameter	Min	Minimum key life in hours
Parameter	Max	Maximum key life in hours

NumComputersOnNetwork	The quantity of computers on a given network.	
Parameter	Name (Displayed in tool)	Description
ConditionText	Network	Network name
Parameter	Min	Minimum number of connected computers to make this true
Parameter	Max	Maximum number of connected computers to make this true

NumLinkEncryptOnNetwork	The quantity of link encryptors on a given network.	
Parameter	Name (Displayed in tool)	Description
ConditionText	Network	Network name

Parameter	Min	Minimum number of link encryptors to make this true
Parameter	Max	Maximum number of connected link encryptors to make this true

NumVPNOnNetwork	The quantity of VPN gateways on a given network.	
Parameter	Name (Displayed in tool)	Description
ConditionText	Network	Network name
Parameter	Min	Minimum number of VPNs to make this true
Parameter	Max	Maximum number of connected VPNs to make this true

ObjectiveCompleted	Whether a given objected has been complted.	
Parameter	Name (Displayed in tool)	Description
ConditionText	Objective	name of the objective

PhaseCompleted	Whether a given phase has been completed. Trigger conditional parameters are compared against the phase number, starting at zero.	
Parameter	Name (Displayed in tool)	Description
ConditionText	"Phase Name"	name of the phase

RandomValue	Generate a random integer between zero and a given number. Random values are assigned when SetRandom triggers fire. See the discussion above about random values.	
Parameter	Name (Displayed in tool)	Description
Parameter	Max	Random number will between zero and this, inclusive.

Register	State variable used with question triggers. Note, register are only intended for use with trigger condition parameters.	
Parameter	Name (Displayed in tool)	Description

ConditionText	name[1]	Name of the register. Initial value is zero. When named in a question trigger, the value becomes the character the player types, e.g., "y", or "c".

StaffMeetsUser	Whether a given staff member is currently interacting with a given user	
Parameter	Name (Displayed in tool)	Description
ConditionText	StaffName	Name of staff member
SecondConditionText	DestUser	Name of user

StaffAtAssetComputer	Whether a given staff member is currently at the computer containing the given asset.	
Parameter	Name (Displayed in tool)	Description
ConditionText	StaffName	Name of staff member
SecondConditionText	Asset	Name of asset

TimeCondition	Whether a given amount of time has passed.	
Parameter (id)	Name (Displayed in tool)	Description
Parameter	ElaspedTime[1]	The number of hours.
Parameter	OneIfPerPhase	0 = from start of game 1 = from start of phase

TriggerGoneOff	The number of times a give trigger has gone off. **Note when assessing trigger condition values, this is a "greater than" comparison.**	
Parameter	Name (Displayed in tool)	Description
ConditionText	TriggerName	Trigger
Parameter	"min #"	minimum
Parameter	"max #"	max

UnaccountableAssetAccess	Test whether a user can access an asset without being identified (e.g., via a workstation that does not require authentication.)	
Parameter	Name (Displayed in tool)	Description
ConditionText	AssetName	The asset

SecondConditionText	User	The user who achieves unaccountable access. If blank, the condition is true if any user has unaccountable access.

UnaccountableWSAccess	Test whether a user's workstation can be accessed by some other user without authentication. Includes a test of (no motive) zone access.	
Parameter	Name (Displayed in tool)	Description
ConditionText	UserName	The user. A blank means any user that has an assigned workstation.

UserAcessToAG	Whether a user can achieve a given goal, and the hassle involved.	
Parameter	Name (Displayed in tool)	Description
ConditionText	User	User name
SecondConditionText	AssetGoal	Goal name
Parameter	0to4	minimum access 0 = full access 1 = via network 2 = not my computer, same zone 3 = must leave my zone 4 = no access
Parameter	0to4	maximum access per above

UserAnimation	Whether the given user's current animation is a given value.	
Parameter	Name (Displayed in tool)	Description
ConditionText	UserName	User

Parameter	Animation	Animation value as follows: SIT 1 STAND 2 TYPE 3 IDLE 4 STARTWALK 5 ENDWALK1 6 SHUTDOWN 7 BLOCKED 8 LOCKEDOUT 9 SLEEP1 10 WALK1 11 WALK2 12 ENDWALK2 13 FRUSTRATED 14 WAITING 15 THREATEN 16 SHAKE_NO 17 IDLE2 18

UserAssignedZone	Whether the user is assigned to a given zone.	
Parameter	Name (Displayed in tool)	Description
ConditionText	User	User of the user
SecondConditionText	Zone	Name of the zone

UserCheckpointAnim	Whether the given user is approaching a checkpoint having given equipment	
Parameter	Name (Displayed in tool)	Description
ConditionText	UserName	User
Parameter	Equipment	Blank (or negative) for any equipment. Otherwise: See shapes.h enumeration

UserFailsGoal	A given user goal has been not achieved over a given period of time.	
Parameter	Name (Displayed in tool)	Description
ConditionText	UserName	Name of a user. A blank means any user.
SecondConditionText	AssetGoal	Name of the asset goal not achieved.

UserHappiness	A given user's happiness has a given value.	
Parameter	Name (Displayed in tool)	Description

ConditionText	UserName	The user name to be tested. Can be the wildcard "*" to denote any user.
Parameter	relation0forLessThan	0 means less than test, 1 means greater than test.
Parameter	TestValue	The amount of happiness to be tested against the user's current happiness using the given relation.

UserHasAssignedComputer	Whether the given user has an assigned computer	
Parameter	Name (Displayed in tool)	Description
ConditionText	UserName	User

UserInZone	Whether a given user (animated character) is currently within a name zone.	
Parameter	Name (Displayed in tool)	Description
ConditionText	User	User name
SecondConditionText	Zone	The zone name

UserMeetsUser	Whether a given user is interacting with another given user	
Parameter	Name (Displayed in tool)	Description
ConditionText	UserName	Name of the first user
SecondConditionText	DestUser	Name of second user

UserProductivity	A given user's productivity has a given value.	
Parameter	Name (Displayed in tool)	Description
ConditionText	UserName	The user name to test. Can be the wildcard "*" to denote any user.
Parameter	relation0forLessThan	0 means less than test, 1 means greater than test.
Parameter	Value	The amount of productivity to be tested against the user's current productivity using the given relation.

UserTraining	Amount of training for a given user	
Parameter	Name (Displayed in tool)	Description
ConditionText	UserName	User

Parameter	Min	minimum training
Parameter	Max	maximum training

UserTrust	Amount of trust for a given user based on background checks and the user's initial trust value.	
Parameter	Name (Displayed in tool)	Description
ConditionText	UserName	User
Parameter	Min	minimum trust
Parameter	Max	maximum trust

UserWsHasIdDevice	Whether the workstation assigned to a given user has an associated ID Device, and if so, its type. Trigger condition parameters assess to the type.	
Parameter	Name (Displayed in tool)	Description
ConditionText	User	User name
Parameter	Type	The ID device type (see ZoneHasEquip for values)

VirusPresent	Whether a virus or Trojan horse is present on a computer assigned to a given user (or optionally any computer).	
Parameter	Name (Displayed in tool)	Description
ConditionText	UserName	Name of user whose computer is to be checked. If blank, all computers are checked.

VirusPresentAsset	Whether a virus or Trojan horse is present on a computer containing a given asset.	
Parameter	Name (Displayed in tool)	Description
ConditionText	Asset	Name of the asset whose associated computer is to be checked.

WsHasIdDevice	Whether the named workstation has an associated ID Device, and if so, its type. Trigger condition parameters assess to the type. Intended for use with static or "no scrap" workstations.	
Parameter	Name (Displayed in tool)	Description
ConditionText	ComputerName	Name of the workstation

Parameter	Type	The ID device type (see ZoneHasEquip for values)

WsUserHasValProfile	Whether the named workstation has a validation profile that permits access by the named user. Intended for use with static or "no scrap" workstations because it uses the workstation name.	
Parameter	Name (Displayed in tool)	Description
ConditionText	ComputerName	Name of the workstation
SecondConditionText	UserName	Name of the user.

ZoneClearanceAccess	Whether a given zone can (should) be entered by someone with a given clearance.	
Parameter	Name (Displayed in tool)	Description
ConditionText	Zone	name of zone
SecondConditionText	Secrecy	Secrecy label of clearance.

ZoneFailsGoal	Whether a zone has failed a zone goal. See the zone form.	
Parameter	Name (Displayed in tool)	Description
ConditionText	Zone	name of zone
SecondConditionText	Goal	Name of the goal

ZoneGroupAccess	Whether a given zone can (should) be entered by someone who is a member of a given group.	
Parameter	Name (Displayed in tool)	Description
ConditionText	Zone	name of zone
SecondConditionText	Group	Name of the group

ZoneHasEquip	Whether a given zone has a type of equipment controlling physical access	
Parameter	Name (Displayed in tool)	Description
ConditionText	Zone	name of zone
Parameter	EquipType	Anything -1 Eye Scanner 3 Hand Scanner 4 Card Reader 5 Eye & Card 6 Hand & Card 7 Authorization Card 8

ZoneHasPolicy	Whether a given zone has a given policy	
Parameter	Name (Displayed in tool)	Description
ConditionText	Zone	name of zone
SecondConditionText	PolicyOfZone	see AssetZoneHasPolicy

ZoneHasSecurityValue	The amount of physical security in a zone.	
Parameter	Name (Displayed in tool)	Description
ConditionText	Zone	Zone
Parameter	Min	minimum
Parater	Max	maximum

ZoneUserAccess	Whether a given zone can (should) be entered by a given user.	
Parameter	Name (Displayed in tool)	Description
ConditionText	Zone	name of zone
SecondConditionText	User	Name of the user

Generating Camera Viewpoints

Make sure the SDT Tools / Project Settings / Runtime Parameters field includes the "-A" flag. Run your scenario and navigate to each desired camera viewpoint in turn and press the "[" key at each desired camera position. Then exit the game and go to the CyberCIEGE/results/your name/log directory and open the viewpoints.txt file. Cut and paste the content of this file into the "Extra Syntax" field of the Scenario Form, adding to, or replacing the current content. These viewpoints can be named in the SetCameraToIndex trigger and the SpeakTrigger (index starting at zero[8]). When players press the <Tab> key, they will cycle through these viewpoints unless a "SkipTab: true :end" line is added to a viewpoint entry. This is useful when scripted camera positions are desired but you don't want these to clutter the player's ability to quickly cycle through the camera positions. You may also add a "Site: site_name :end" entry where site_name is the name of a zone that is either the entire office, or an offsite zone. These viewpoint are then used when the player tabs to the next building.

Attack Types

Attacks are generated by AssetAttack triggers. The types of attacks include:

Attack Description	**Attack ID**
Bad policies resulting in malware installation. E.g., allowing external un-vetted software. Note the "motive" on these attacks differs from the other attacks because this does not directly attack assets. If malware installation succeeds, then vulnerable computers receive a Trojan horse and a virus. However, if the motive is -4, it will only receive a Trojan horse. And -5 will only result in a virus. The latter degrades the component's availability. Both can lead to attacks on assets through "Targeted malicious software attacks" described below.	7
Malware propagation in which malware introduced via (7) spreads itself to other computers. Motive has no effect on whether malware spreads.	20
Break-in and hacking. The attacker will enter a zone if possible and either directly access computers or hook a portable computer to a LAN or indirectly access an asset via some connected computer. These attacks do not include Internet attacks	18
Attacks on assets from the Internet. These do not include wiretap attacks.	19
Insider Hacking. These are identical to the Break-in and hacking attacks (18) except they are performed by insiders whose trust is less than the motive.	17

[8] Note triggers index viewpoints starting at zero, but when players press keyboard numbers the indexing starts at one.

Insider who has authorized access to an asset is bribed to compromise the asset.	16
Targeted malicious software is used to compromise an asset. This software might be installed due to bad policies (7), or it might simply come from the factory that way. The integrity of operating systems and applications drive these attacks.	8
Wiretap to read or modify the bits on a wire. These attacks do not involve hacking into components (that would be 18 or 16)[9]. However these attacks may include elaborate spoofs (e.g., standing up a fake web site and altering the network routing to direct users to the fake site).	9
Denial of Service attacks from the Internet - Successful DoS will degrade availability of affected computers. These attacks depend on unpatched flaws in software	15
Denial of Service flood attacks – symptoms and behavior is similar to (15), however they do not rely on unpatched flaws. The engine looks for filtered software types in goals that access the given asset. Attacks succeed if the software type can reach the component from the internet.	21
Equipment failures	14
Smart lock attacks – smart locks with card readers provide strong security. However, use of smart locks can increase attacker motive to compromise the databases used when issuing smart locks.	1
Spoofed site attacks. User might follow an email link or a web link to a bogus site on the internet. While there, the user might get a false view of high integrity data (reference integrity), or the user might be fooled into entering sensitive data, which is captured by the bogus site. These are similar to wiretap attacks on the Internet, however achieving the user goal need not traverse the Internet. These attacks only occur on goals requiring web services.	2
Spoofed email attacks. An attacker (or suitably motivated insider) creates bogus email in the stead of an email asset having an integrity motive. The attacker need only be able to send email to the server containing the actual email asset. These attacks are countered by the sender signing the email and the receiver authenticating the email and its associated certificates.	10

[9] A wiretap might include hacking or otherwise subverting network naming infrastructure (e.g., DNS) that is not directly represented within the game.

Attack Properties

These attack properties appear in AssetAttacked conditions and AssetToNetwork conditions. They also can be used to control computer diagnostic messages resulting from ComputerDiagnostics triggers. Note the ID values are actually powers of two, and thus can be ORed together.

Attack or Problem	**ID**
Password sniffed via malware on workstation.	0
The computer does not require a password	1
Attacker directly attacked the computer via its console	2
Component compromised via a weak password	3
Component compromised because a weak ID Device was used to identify the user	4
A VPN gateway or a VPN client was subverted	5
A VPN client was bypassed via island hopping	6
A VPN gateway was bypassed via split tunneling	7
The VPN client requires a measured boot, but the computer is not configured to require it.	8
A component contains malware	9
Weak card stripe:	10
Subverted O/S:	11
O/S Flaw	12
No O/S protection for asset	13
Reference integrity	14
Wire tap	15
Unpatched services exposed to the internet	16
Subverted network service	17
Access is permitted	18
Unattended workstation left logged on	19
Subverted boot (e.g., via attacker's media)	20
Access to the component permitted	21
Network filters flaw (e.g., in platform or filters applications)	22
Loose network filter	23
The PKI mechanism has no installed root	24
The PKI mechanism has no key/certificate (no CA selected)	25
VPN lacks a protected network selection	26
A public pay-per-cert CA issued bogus certificates	27
An enterprise CA issued bogus certificates	28
Stale symmetric keys used and found on vulnerable vpn	29
Malware on sender's workstation compromised email	30
Malware on receiver's workstation comprised email.	31
Motive stronger than the crypto algorithm	32
Missing smartcard reader	33
Email not protected	34

A partner's Certification Authority was compromised and issued a bogus cert	35
A partner's CA issued a bogus certificate and cross certification led to reliance on such a certificate	36
The attacker was permitted access to the zone in which the compromise occurred.	37
An indirect attack used a computer as a gateway to hop from one network to another.	38
An attack occurred using a compromised or unprotected web server	39
An attacked occurred because a workstation user has administrator rights, or a server's users all have root access.	40
The receiver of an email failed to authenticate the message and was spoofed.	41
A cryptographic key was disclosed	42
A smartcard was used as an illicit communication path between two computers.	43
A smart lock was used to protect a valuable asset and the database controlling issuance of smart cards was compromised due to the increased motive. Note this property is assigned to the attack on the ID database and to the attack on the assets protected by the smart lock.	44
A VPN client is configured to protect the data, but there is no remote VPN mechanism with which to communicate	45
User browser's procedural settings do not require use of SSL when achieving this goal.	46
Attacker used a valid certificate properly issued by some CA during the attack.	47
User procedures allow the user to accept any root or server certificate.	48
Multilevel channel is connected to a computer that does not support multilevel channels	49
Denial of Service	50
Mandatory access control mechanism defeated	51
Signing key utilized for TLS was used to sign bogus email	52
Denial of service flood attack	53
Compromise of asset resulting from the KeyExposed trigger	54

Operating System Assurance

In the following table, assurance values determine an ability to enforce policies. The integrity value is with respect to software integrity vice policy enforcement (can be seen as assurance the software is free of Trojan horses.)

O/S	DAC Assurance	MAC Assurance	Integrity
Populos V9 Desktop	60		150
Trusted Populos Desktop	EAL4	EAL4	500
Lunitos Desktop	80		150

JarLid Desktop	90		150
Green Shade Core	--	EAL7	--
Populos V9 Server	60		60
Secure Shade Desktop	EAL7	EAL4	500
Populos V8 Server	EAL4		150
Trusted Populos Server	EAL4	EAL4	500
Secure Shade Server	EAL7	EAL7	500
Jar Lid Server	80		200
Populos Embedded V5	60		150
FlipOS	80		
FlipStrip	210		
MergerTech SOS	60		200
GEOS	900		900
CREOS	600		600
GIN	600		600

Valid O/S – Hardware Combinations

The SDT O/S pulldown lists are a function of the selected hardware. This is maintained in the hwOs.ini as well as the CyberCIEGE source code.

Valid Software – O/S Combinations

The SDT software list is a function of the selected O/S. This is maintained in the swOS.ini file as well as the CyberCIEGE source code.

User Thought Priorities

static UserThought_t _thoughts[THOUGHT_MAX_TYPES] = {
{ 0, NULL, NULL, "I like working here.", THOUGHT_HAPPY, THOUGHT_HAPPY_WORK, 3 },
{ 0, NULL, _userThoughtCheckWebBrowser, "I like to surf the web.", THOUGHT_HAPPY, THOUGHT_HAPPY_WEBSURF, 3 },
{ 0, NULL, NULL, "How did I get this cushy job?", THOUGHT_HAPPY, THOUGHT_HAPPY_CUSHYJOB, 3 },
{ 0, NULL, NULL, "It could be worse I guess.", THOUGHT_NEUTRAL, THOUGHT_NEUTRAL_BEWORSE, 3 },
{ 0, NULL, NULL, "Yeah, whatever.", THOUGHT_NEUTRAL, THOUGHT_NEUTRAL_WHATEVER, 3 },
{ 0, NULL, NULL, "Another day, another dollar.", THOUGHT_NEUTRAL, THOUGHT_NEUTRAL_DAYDOLLAR, 3 },

```
    { 0, NULL, NULL, "I wish we had better computers.", THOUGHT_FLAVOR,
THOUGHT_FLAVOR_BETTER_MACHINES, 3 },
    { 0, NULL, NULL, "The mouse is my friend.", THOUGHT_FLAVOR,
THOUGHT_FLAVOR_MOUSE_FRIEND, 3 },
    { 0, NULL, NULL, "I want to go to lunch.", THOUGHT_FLAVOR,
THOUGHT_FLAVOR_LUNCH, 3 },
    { 0, NULL, NULL, "I wish I were paid more.",     THOUGHT_FLAVOR,
THOUGHT_FLAVOR_PAIDMORE, 3 },
    { 0, NULL, NULL, "Will I be fired?", THOUGHT_FLAVOR,
THOUGHT_FLAVOR_FEAR_FIRED, 3 },
    { 0, NULL, NULL, "Will I be promoted?", THOUGHT_FLAVOR,
THOUGHT_FLAVOR_HOPE_PROMOTE, 3 },
    { 0, NULL, _userThoughtCheckComputer, "My machine is down.",
THOUGHT_SUPPORT, THOUGHT_SUPPORT_MACHINE_DOWN, 15 },
    { 0, NULL, _userThoughtCheckNetwork, "The network is so slow.",
THOUGHT_SUPPORT, THOUGHT_SUPPORT_NETWORK_SLOW, 15 },
    { 0, NULL, _userThoughtCheckComputer, "My machine keeps rebooting.",
THOUGHT_SUPPORT, THOUGHT_SUPPORT_MACHINE_REBOOT, 15 },
    { 0, NULL, _userThoughtCheckSoftware, "My software doesn't run.",
THOUGHT_SUPPORT, THOUGHT_SUPPORT_SOFTWARE_FAIL, 15 },
    { 0, NULL, _userThoughtCheckComputer, "My computer keeps crashing.",
THOUGHT_SUPPORT, THOUGHT_SUPPORT_MACHINE_CRASH, 15 },
    { 0, NULL, NULL, "I can't meet my #a goal", THOUGHT_ACCESS,
THOUGHT_ACCESS_ASSET, 100 },
    { 0, NULL, _userThoughtCheckNetwork, "I can't get onto the network.",
THOUGHT_ACCESS, THOUGHT_ACCESS_NETWORKFAIL, 15 },
    { 0, NULL, NULL, "Where is that file?", THOUGHT_ACCESS,
THOUGHT_ACCESS_FILELOST, 3 },
    { 0, NULL, NULL, "I need #s for my machine.", THOUGHT_ACCESS,
THOUGHT_SUPPORT_WANT_SOFTWARE, 80 },
    { 0, NULL, NULL, "I don't have rights to access the server.",
THOUGHT_ACCESS, THOUGHT_ACCESS_RIGHTS, 3 },
    { 0, NULL, NULL, "IT is our enemy.", THOUGHT_ACCESS,
THOUGHT_ACCESS_HATE_IT, 3 },
    { 0, NULL, NULL, "I hate working here.", THOUGHT_COMPLAINT,
THOUGHT_COMPLAINT_WORK, 3 },
    { 0, NULL, NULL, "I wish I were in Utah", THOUGHT_COMPLAINT,
THOUGHT_COMPLAINT_UTAH, 3 },
    { 0, NULL, NULL, "I can't wait to go home.", THOUGHT_COMPLAINT,
THOUGHT_COMPLAINT_GO_HOME, 3 },
    { 0, NULL, NULL, "IT is run by fascists.",   THOUGHT_COMPLAINT,
THOUGHT_COMPLAINT_IT_FASCISTS, 3 },
    { 0, NULL, _userThoughtCheckNetwork, "I wish we had a faster network
connection.",  THOUGHT_COMPLAINT,
THOUGHT_COMPLAINT_NETWORK_SLOW, 3 },
```

{ 0, NULL, _userThoughtCheckComputer, "I created an asset", THOUGHT_MAKE_ASSET, THOUGHT_MAKE_ASSET_COPY, 3 },

{ 0, NULL, _userThoughtCheckGuard, "I wonder if the guard's gun is loaded.", THOUGHT_POLICY, THOUGHT_POLICY_GUARD, 6 },
{ 0, NULL, _userThoughtCheckGuard, "That guard just walks around all day.", THOUGHT_POLICY, THOUGHT_POLICY_PATROL, 8 },
{ 0, NULL, _userThoughtCheckGuard, "I hate having security looking at me.", THOUGHT_POLICY, THOUGHT_POLICY_VISUAL, 8 },
{ 0, NULL, _userThoughtCheckPhones, "I really hate not having my cell phone here.", THOUGHT_POLICY, THOUGHT_POLICY_PHONE, 9 },
{ 0, NULL, _userThoughtCheckCamera, "I'm gonna smash that stupid camera one day.", THOUGHT_POLICY, THOUGHT_POLICY_CAMERA, 9 },
{ 0, NULL, _userThoughtCheckXRay, "I really hate having my stuff X-Rayed.", THOUGHT_POLICY, THOUGHT_POLICY_XRAY, 9 },
{ 0, NULL, NULL, "I think Iris Scanners are annoying.", THOUGHT_POLICY, THOUGHT_POLICY_IRIS1, 11 },
{ 0, NULL, _userThoughtCheckBadge, "A badge, yet another nuisance.", THOUGHT_POLICY, THOUGHT_POLICY_BADGE, 7 },
{ 0, NULL, NULL, "I think IT should do backups, not me.", THOUGHT_POLICY, THOUGHT_POLICY_BACKUP1, 5 },
{ 0, NULL, NULL, "I don't like machines without floppys.", THOUGHT_POLICY, THOUGHT_POLICY_MEDIA1, 5 },
{ 0, NULL, _userThoughtCheckConfigMngmt, "Config Management, blah.", THOUGHT_POLICY, THOUGHT_POLICY_CM1, 5 },
{ 0, NULL, _userThoughtCheckPatches, "I don't like apply patches.", THOUGHT_POLICY, THOUGHT_POLICY_PATCHES, 5 },
{ 0, NULL, _userThoughtCheckReceptionist, "I hate that receptionist.", THOUGHT_POLICY, THOUGHT_POLICY_RECEPTIONIST_HATE, 5 },
{ 0, NULL, _userThoughtCheckReceptionist, "I like that receptionist.", THOUGHT_POLICY, THOUGHT_POLICY_RECEPTIONIST_LIKE, 5 },
{ 0, NULL, _userThoughtCheckEmailStrip, "I don't like my email attachments stripped.", THOUGHT_POLICY, THOUGHT_POLICY_EMAIL_STRIP, 5 },
{ 0, NULL, _userThoughtCheckEmailAttachExecute, "Why can't I execute my email attachment?", THOUGHT_POLICY, THOUGHT_POLICY_EMAIL_ATTACH_EXECUTE, 5 },
{ 0, NULL, _userThoughtCheckWebEmail, "Why can't I read my web email account?", THOUGHT_POLICY, THOUGHT_POLICY_WEB_EMAIL, 5 },
{ 0, NULL, NULL, "I don't like #p", THOUGHT_POLICY, THOUGHT_POLICY_GENERIC, 2 },

{ 0, NULL, NULL, "Someone stole my computer!!!", THOUGHT_ATTACK, THOUGHT_ATTACK_STEALCOMP, 50 },
{ 0, NULL, NULL, "Someone stole my device!!", THOUGHT_ATTACK, THOUGHT_ATTACK_STEALDEV, 45 },

```
	{ 0, NULL, NULL, "Uh oh, was it me?", THOUGHT_ATTACK,
THOUGHT_ATTACK_STEALMEDIA, 30 },
	{ 0, NULL, NULL, "Why would IT ask me for my password?",
THOUGHT_ATTACK, THOUGHT_ATTACK_SOCIAL, 30 },
	{ 0, NULL, NULL, "Someone's calling to ask for my password?",
THOUGHT_ATTACK, THOUGHT_ATTACK_SOCIAL2, 30 },
	{ 0, NULL, NULL, "This has got to be a scam...", THOUGHT_ATTACK,
THOUGHT_ATTACK_SOCIAL3, 30 },
	{ 0, NULL, NULL, "I wonder why they asked for this data...",
THOUGHT_ATTACK, THOUGHT_ATTACK_SOCIAL4, 30 },
	{ 0, NULL, NULL, "Mmm, my chair is warm.", THOUGHT_ATTACK,
THOUGHT_ATTACK_WARMCHAIR, 10 },
	{ 0, NULL, NULL, "Should I have done that?", THOUGHT_ATTACK,
THOUGHT_ATTACK_SHOULDI, 20 },
	{ 0, NULL, NULL, "I've got a Virus!", THOUGHT_ATTACK,
THOUGHT_ATTACK_HAVEVIRUS, 100 },
	{ 0, NULL, NULL, "I've discovered a Virus Attempt!", THOUGHT_ATTACK,
THOUGHT_ATTACK_FOUNDVIRUS, 100 },
	{ 0, NULL, NULL, "Please install our special software? OK.",
THOUGHT_ATTACK, THOUGHT_INSTALL_TROJAN, 30 },
	{ 0, NULL, NULL, "Please install our special software? No!.",
THOUGHT_ATTACK, THOUGHT_NOINSTALL_TROJAN, 29 },
	{ 0, NULL, NULL, "I think I'll install #s.", THOUGHT_ATTACK,
THOUGHT_INSTALL_UNAUTHORIZED, 10 },
};
```

Troubleshooting and Problems

If the SDT seems stuck on one project and will not let you change projects you may have to delete the datapath.ini file that is usually in C:\Documents and Settings\[your login name]\CyberCIEGE.

Catalog Components
When you define catalog components in your sdt, take care to group all workstations together; all servers together; and all network devices together. Otherwise the game gets very confused and will not display all your catalog items.

If help tips lack the triangle, try moving the box up or down a bit.

User the "Move Up" button to re-order elements of a set.

If you define multiple sets, the order still matters. Here you can arrange the order by removing all sets from the scenario and then adding them in the desired order.

Triggers
Trigger order matters. If multiple triggers go off at the same time, the engine generally selects the first one in the SDT. Attack response trigger order matters, e.g., if one trigger is only supposed to fire after another has fired a set number of times.

Briefing Text
Sometimes the gui cuts off the end of a briefing, so you may want to end each briefing text with (PARAGRAPH) to prevent truncated text.

Web Servers
An asset having a filtered goal must appear on a web server for the server to be configurable as a web server. And to work right, goals must include browsers.

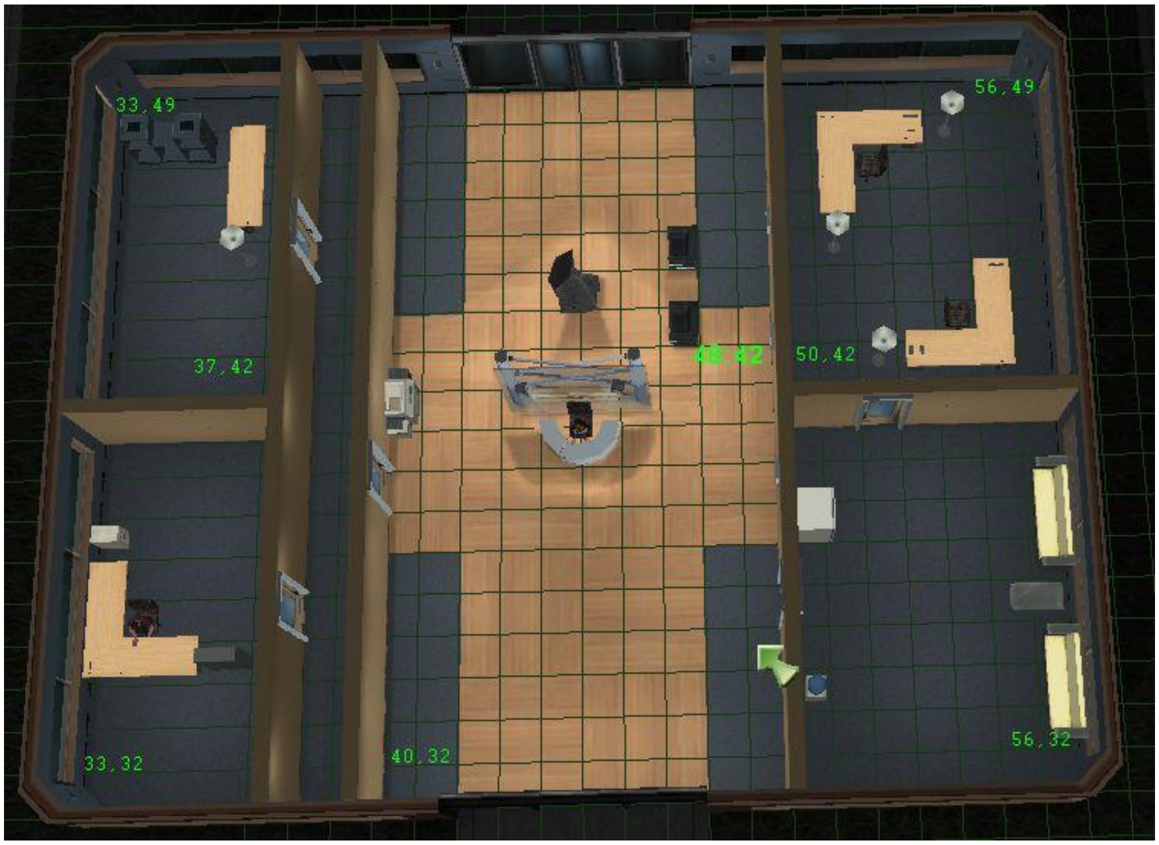


Figure 3: Main Office Grid Layout

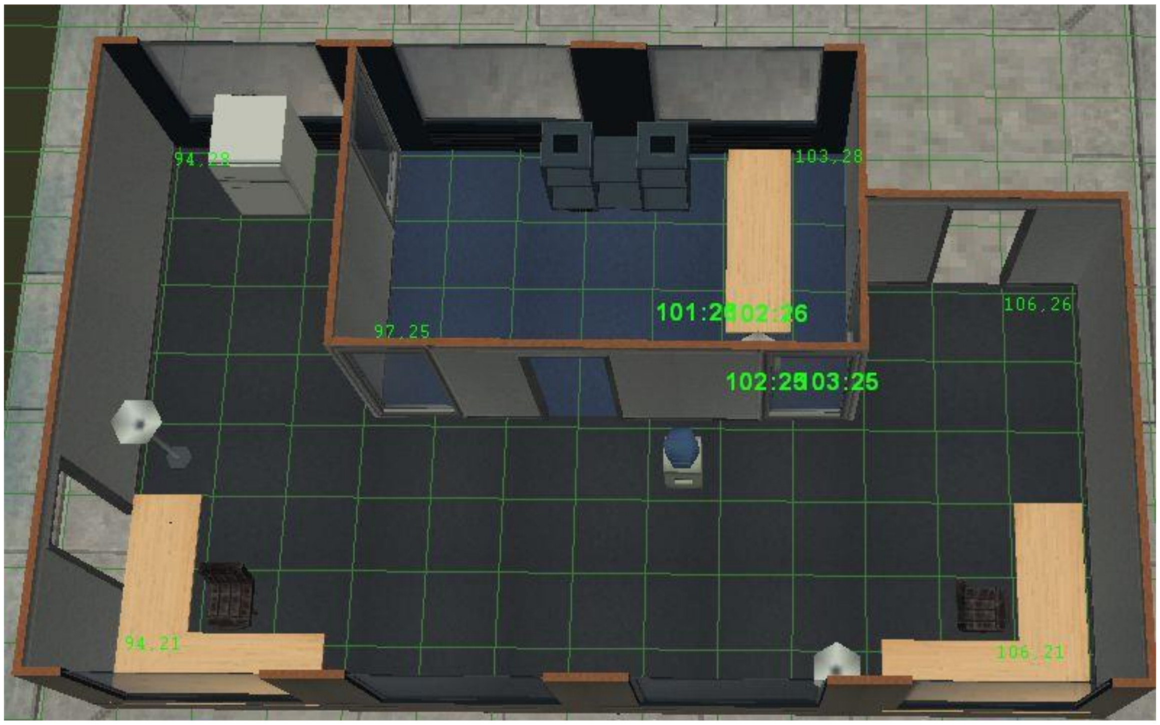


Figure 4: Offsite Office Grid Layout

Conditions, Triggers and other Configurable Data

The conitions.ini and triggers.ini files allow the addition of new conditions and triggers without code changes. Reference the headers of those files for a description of the syntax.

Adding Game Objects

A few shapes are available as objects that can be added to the 3D environment with the ShowObject trigger. The most common is shape "1", which is a billboard. The content of the billboard is controlled via the ObjectTexture trigger, which names a tga file in the game/exec directory. Images are limited to 128x64 pixels and they should be formatted without compression.

Appendix A: Notes

A.1 Reference Integrity

The simple simulation model associates secrecy and integrity motives with assets. These motives dictate the types and strengths of attacks on assets. The model does not deal with copies of assets or the flow of one asset to another[10]. The model simply deals with ways in which an asset might be improperly disclosed or modified (and that includes wiretaps of assets that traverse networks as part of user goals.)

That simple model comes up a bit short when representing the need to maintain the integrity of a user's view of high integrity information. An example is the need to reference a remote read-only high integrity biometric database. The biometric database asset may be thoroughly protected from unauthorized modification, but the user may see a spoofed view of the data (e.g., due to someone injecting data on the wire).

When determining the motive for an attacker to present a user with a spoofed view of a high integrity asset, we cannot simply use the motive of the asset. This is because some other user may have a low integrity need to read the same asset, i.e., a need to read the data from a low integrity environment for a non-critical purpose.

Accounting for this in the model required a small change: we simply include an optional motive & cost with the user goals. The extended semantics of a user goal from the perspective of the scenario designer is: "This goal is to read that asset. Attackers have a motive of Y to present the user with a spoofed view of the asset, and if the user sees a spoofed version of that asset, the cost is X." An example of this is phase 3 of the IdM scenario. In that scenario, it is not the remote database that is corrupt, the biometric scanner's view of the asset is spoofed.

Note that there is no need to add motives and costs to goals for secrecy. This is because people don't generally have a "write only" need to access data (of course processes may, but our model is of users and information). For example, if we wanted to model the need for a user to remotely add low secrecy data to a high secrecy asset from a low-secrecy location, then we'd need to add a goal motive and cost for secrecy -- but such a thing is not high on our list of missing fidelity. And thus, the motive to disclose an asset is always simply the motive of the asset.

In a nutshell, the educational point is this: When determining the protections needed to allow user's to securely access assets, we should consider an attacker's motive to compromise the asset. However, some uses of an asset might require less integrity than is otherwise required in the protection of the asset itself. And thus some users might

[10] An exception is the processing of email attacks in which temporary copies of plain-text email is placed on the sender and receiver's workstations and then subjected to the attack engine.

reasonably read high integrity assets from relatively low integrity environments[11]. The same is generally not true for protecting the secrecy of assets. An attacker's motive for disclosing an asset should always dictate the protections provided when users access the asset regardless of the user's intended use of the asset.

Within the game, reference integrity attacks occur as a part of wiretap attacks. They also occur as part of “spoofed site” attacks in which users are lured into visiting bogus instances of sites. These attack types are distinguished by not relying on attacker access to networks traversed when users achieve goals. If the user has Internet access, spoofed site attacks may succeed (unless countered via SSL or VPNs).

Note that for now, email is destined for a single user, so we can use its integrity to represent an attacker’s motive to spoof a user.

Appendix B: Attack Engine Notes

Someday resources will appear to allow us to create a systematic description of the game’s attack engine and its various caveats and approximations. Until that day:

- Motive must be greater than assurance (or protection) for the attack to succeed.

- Trojan horses start to appear within existing software at motive > 50 and Trojan horses don’t exfiltrate explicit assets until motive is 50.

- Subverted network services act as Trojan horse with motive > 40, letting attacker browse for documents.

- Wiretap by outsider if motive > 30, that includes session hijacking.

- Wiretap by insider using assigned computer with motive >30; Otherwise motive >55

- Computers can be used gateways based on the protection provided the computer, but minimum motive of 35 is needed.

- Patch management can cause weakness in computers offering services. If computer contains software that provides internet service and computer not patched, results in virus per bad policies logic. Software assurance is halved if computer is unpatched and that software type is available to the Internet.

- When network filters are deployed, assets can still be compromised via network connections depending on which ports are open. The engine looks at any services

[11] In a true technical security policy, this would be represented by the subject of the reference integrity modifying an object of the same reference integrity. The protection mechanism must ensure the subject cannot read lesser integrity (e.g., cannot be spoofed into reading a bogus version of a high integrity object.) The game’s use of reference integrity lets us avoid introducing additional objects (assets).

(i.e., "filterable" software) that are used as part of an asset goal to access resident assets. If a service is subverted, and the corresponding port is open, compromise can occur. Also, if the component allows remote access to the attacker (or has weak password), any services on the component can be used to compromise the asset – whether or not the service is part of a goal.

- Filters will block access to compromised software services – or to services on machines that have no remote authentication.

- Filters are considered "loose" if they have three or more open ports besides Web, email and SSH services.

- If filter exceptions are made for domains that exist within a WWW zone, it is assumed the attacker's computer can originate from that zone.

- Attacker direct access to computer results in subverted boot if motive > 50 and assurance less than 400 (for MAC computers) or less than 900 for non-MAC.

- At motives of at least 80, routing mechanisms might become subverted such that a symmetric key VPN mechanism will be spoofed into communicating with a remote VPN gateway that is not the one that it expects. This attack assumes the use of "stale" symmetric keys that are present on the two intended VPN mechanisms as well as the rouge VPN mechanism.

- Island hoping w/ VPN gateways (either via use of unprotected connection profiles or side-band routers) require the attackers computer to be recognized by the VPN Gateway as a legitimate local workstation. VPN Gateways only name local hosts via spoofable IP addresses. Thus, any subvertable component on the VPN Gateway's LAN connection can proxy for the attacker's computer

- CA assurance is based on the assurance of the underlying OS and the CA application. Public CAs are always vulnerable and a motive higher than the CA assurance will defeat them. Partner CA vulnerability can be increased by poor CM (e.g, player can see the partner has poor CM.) Enterprise CAs are attacked using standard attack logic using the motive of the asset being protected.

- Wiretaps are modeled by putting assets on wires as part of goal assessment. If an asset is put on a wire as part of a shared goal in which both parties only used crypto-enabled software (e.g., peer-to-peer messaging software), the asset is protected to the assurance of the software. If only the server side (i.e., the side with the asset) has crypto, then protection is offered if filtered software has crypto.

- Spoofed site attacks can result in reference integrity or secrecy compromises. Spoofed site attacks only occur when assets are accessed via the Internet and motive exceeds 30.

- Publically accessible workstations that are offsite are assumed to contain password sniffers, (e.g., via keyboard loggers).

- Use of software services in attacks.
  - Attacks all start at the asset's computer. The attack engine finds shellable services on that and reduces the set of services as is traverses through filters. When encountering computers as gateways, it finds shellable services on each of its ingress networks and treats those as the new set to be reduced by subsequent filters. Each route entry stores its own set of shellable services and these are what are reduced when filters are encountered.

  - Unless subverted, application services will generally only potentially provide attackers with access to assets for which some user has a goal. Web and email goals will generally result in ACLs granting the necessary access to the respective daemons. (NOTE: ACLs that permit a user/group access but not "web service" will still allow the user to access the asset as if the web server used a setuid type of feature and the user is authenticated.)

  - If SSL enabled services are used on SSL-protected assets, the engine alters the service (e.g., web service) to its SSL-equivalent.

  - Software is attacked at start of each cycle and its assurance is adjusted and attack profiles are recorded with the software assurance. If software has "few" patches, then patch management makes a difference. If "moderate" patches, then sw assurance will be halved (intended use of this is to capture event for reporting to the user.)

  - *Remote Authentication* is a big switch that determines if assets on the computer can be potentially accessed remotely by anyone having access to an incoming network. This switch is not a clear analog to real-world settings, rather it is an educational abstraction. If remote authentication is not required, then only ACLs and labels constrain access to assets on the computer. Encryption such as email encryption further constrains access to the content of the asset. Also, the SSL "username / password" requirement overrides the Remote Authentication setting for web-based access to assets.

  - When remote authentication is required, **and** the computer includes a web server application, **and** the asset has intended access of public, **and** the asset is not protected with TLS, then access is permitted.

  - Remote access via authentication-enabled services such as web servers will cause "Set UID" type behavior, i.e., ACLs can be used to limit access to assets. The daemon entries (e.g., "Web Server") are synonymous with "public".

  - When considering back-end systems (e.g., a database behind a dmz), a back-end service will not be compromised if the motive is less than 55 and the service is patched (even though service may have zero days).

  - If remote authentication is not required, but ACLs protect all of the assets, the attacker must either steal a password or subvert a service, i.e., **lack of remote authentication does not imply lack of access control.**

- Software-based integrity attacks will occur from any connected computer having an internet connection and software integrity less than the motive. This simulates instructions coming from the Internet and not the initial source of malware. If no internet connection, then a motive of greater than 500 will cause the software to attack the asset without internet instructions.

- Wiretaps can compromise passwords at motives greater than or equal to 50. With respect to encrypted email, this only shows up when email encryption is used over an unprotected network and the password leads to access to some other asset on the email server. Passwords exchanged between systems that share an authentication server won't be tapped because it is assumed they would have suitable encryption.

- Successful DoS will degrade availability of affected computers. The effects of a successful DoS attack will persist until either a duplicated attack fails or the asset/computer pair is no longer part of any asset goal claim or SMTP claim. Only one open DoS attack on any given asset at any one time, i.e., additional attacks are not recorded. Each asset has a set of computers that have been hobbled by the DoS attack and this is used to reflect the DoS is underway.

- Within the attack logic, there is a special case for physical security that relies on smartcards / biometrics in lieu of security guards. If there is a smart card minter that is used in a goal, and that goal includes a database, then an integrity compromise of that database will reduce the physical security offered by the smart card system. Attacker motive to compromise the smartcard database increase to the motive for assets that are being protected by the smartcards via automated key-locks.

Appendix C: CyberCIEGE Subsystems

CyberCIEGE is a training tool that enhances information assurance education and training through resource management simulation techniques. CyberCIEGE presents students with a three dimensional world where they spend virtual money to operate and defend computer networks. Students see the consequences of their choices including effects on user productivity and network attacks. CyberCIEGE includes the following major functions:

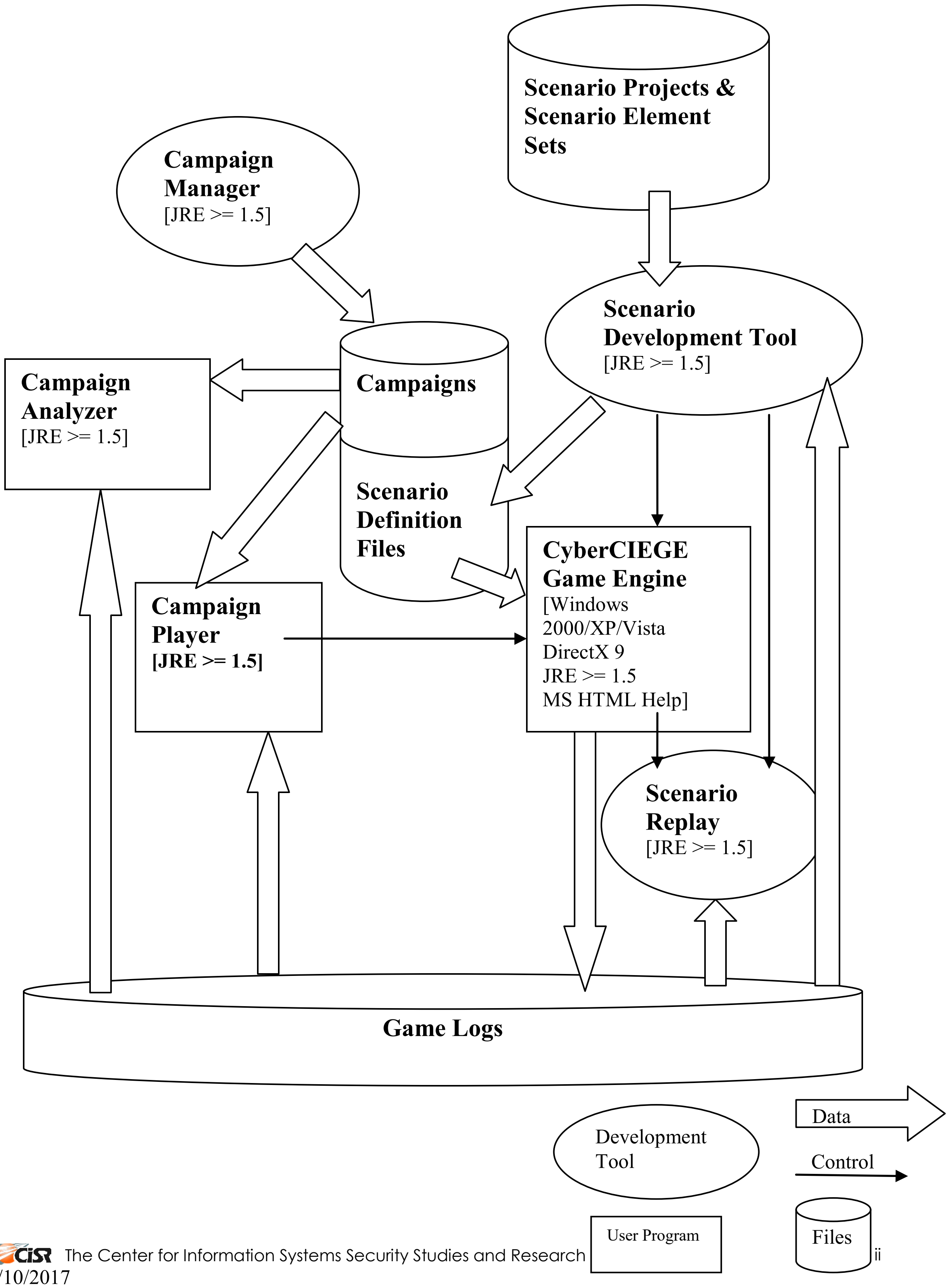
- CyberCIEGE Game Engine
- Campaign Player
- Campaign Analyzer
- Scenario Development Tool
- Campaign Manager

Students use the Campaign Player to select a specific CyberCIEGE scenario to play. The game engine consumes the selected "scenario definition file", and presents the student with the scenario defined therein. DirectX 9 is used to render the 3D world on the student's computer screen. Students use the keyboard and mouse to navigate the virtual world and to purchase and configure network components. Game logic assesses network vulnerabilities and user goal achievement and provides students with corresponding feedback in the form of asset compromises or lost productivity penalties. The game engine creates log files that can be viewed by the student via the Campaign Player, or by an instructor using the Campaign Analyzer. CyberCIEGE also includes an on-line user's guide and help system created using the "Microsoft HTML Help Workshop"; and it includes a series of tutorial animated flash movies that are packaged as executables with embedded flash players.

Scenarios can be customized, or created from scratch using the Scenario Development Tool. And "campaigns" of scenarios can be packaged together using the Campaign Manager.

System requirements include: Windows Vista, XP or Windows 2000; DirectX 9; a video interface with at least 64MB of video RAM; and, the Java runtime environment (1.5 or later).

CyberCIEGE Execution Environment


Scenario Projects & Scenario Element Sets
Campaign Manager [JRE >= 1.5]
Scenario Development Tool [JRE >= 1.5]
Campaign Analyzer [JRE >= 1.5]
Campaigns
Scenario Definition Files
CyberCIEGE Game Engine [Windows 2000/XP/Vista DirectX 9 JRE >= 1.5 MS HTML Help]
Campaign Player [JRE >= 1.5]
Scenario Replay [JRE >= 1.5]
Game Logs
Development Tool
Data
Control
User Program
Files