Fuzzing is a software testing technique that introduces invalid, malformed, or random data to parts of a computer system, such as files, network packets, environment variables, or memory. How the tested item reacts to unexpected data becomes the source of security bugs. Here are a few questions that point to the value of fuzz testing:

- Does the system deal gracefully with unexpected data?
- Does the application crash or fault?
- Does the system enter into an unstable state?

The results of fuzz testing reveal vulnerabilities causing a crash or fault, or other frailty that possibly exposes an application to additional attacks. Fuzz testing exposes security problems in software or computer systems. It uncovers more major issues than other testing methods.

**Peach Fuzzer® Platform**

As a fuzzer, the Peach Fuzzer Platform generates test cases that contain random, bad, or malformed data (or fuzz). The objective of the test cases is to create conditions that uncover security deficiencies in the test subject.

The Peach Platform performs the following tasks:

- **Creates fuzzed data** that tests data values, data relationships, and data flow/state in the system undergoing testing
- **Produces test cases** that contain the mutated data
- **Performs automation** required to run the test case
- **Monitors executing test cases** to identify when a fault has occurred
- **Collects data** about each faulting test case
- **Reports findings**

**How the Peach Platform works**

Three components make up successful fuzzing: modeling, mutating and monitoring.

**Modeling**

The Peach Platform models a data consumer with two components: a data model and a state model. The data model defines each data item by type, size, and occurrence (single or array). The state model addresses data flow and process transitions. The Peach Platform fuzzes both models.

The Peach Platform includes one or more Peach Pits (fuzzing definitions) to target your data consumers. The list of available Peach Pits is quite extensive and includes different file types and common protocols. Here is a sampling of available fuzzing definitions: image formats—PNG, BMP, and JPEG; video format—AVI; and common protocols—Cisco Discovery Protocol, Ethernet, IGMP, IPv4, IPv6, HTTP, and Modbus.

For testers that cannot find the fuzzing definitions they need in Peach’s pre-defined Pits, Peach enables you to create custom Peach Pits to test custom or proprietary protocols, data formats, and systems. This extensibility makes it possible for the Peach Platform to fuzz proprietary internal data formats, network protocols, and embedded systems.
Mutating
The Peach Platform fuzzes the items modeled in the file format or protocol by generating test cases. The platform supplies mutated values for each data item or unusual state transition, using algorithms designed to generate test cases that trigger faults in the target system. The Peach fuzzing engine includes an expansive collection of mutation algorithms and strategies to generate test cases.

For example, look at a four-byte integer, one of the simplest things to test. The Peach mutators generate test cases for the data item in the following ways:

- Supply data values around the maximum and minimum values supported by the number type
- Supply data values around the default or existing value
- Supply data values that coincide with byte boundaries within the number type
- Test the impact of value changes downstream; that is, on related data structures that have a size, count or offset relationship to the number

State fuzzing explores valid and invalid state transitions, conditions needed for a state to occur, and tracking conditions in a state throughout the duration of the state. The Peach mutators generate test cases for state and data flow in the following ways:

- Cause specific actions in a state or entire states to occur more than once
- Perform out-of-order state transitions
- Skip actions in a state or skip entire states

The Peach Platform contains over 40 mutation algorithms for fuzzing arrays, strings, flags, numbers, state transitions and data conditions.

Monitoring
The Peach Platform has a robust monitoring system that encompasses fault detection, data collection, and automation tasks.

Fault detection recognizes when a specific condition or state occurs that is potentially harmful or immediately so, such as a crash. The Peach Platform supports the use of multiple monitors to detect different fault-generating conditions simultaneously. The list of supported monitors includes standard debugging monitors for Windows, Linux, and OS X. Other monitors used in detecting faults watch for conditions or for items such as the following:

- Program crashes
- Error messages in logs
- Error messages via serial console output
- Return codes from scripts that run locally or over SSH
- Unavailable device status
- Unresponsive device behavior

The Peach Platform collects additional data from the test target. In turn, this data contributes in the root cause analysis of faults generated through testing. Common examples of data collection include capturing network traffic, saving log files, and collecting crash dumps (core files).
The Peach Platform can orchestrate the entire fuzzing environment, whether simple or complex, such as automating the testing of external devices. The keys to fully automating the fuzzing environment are to identify and monitor all irregular behaviors, and to reset the environment to a known working state when needed.

The Peach platform includes a number of monitors to assist in automating the environment. A few tasks that are often automated are: controlling power to external devices, running programs locally or remotely, and controlling virtual machines.

**How the Peach Platform reports findings**

The Peach Platform provides results of each fuzzing session in a report that provides an overall summary, as well as several statistical perspectives about the body of information discovered during fuzzing. The platform also provides details about each fault, including a classification of the fault, the risk that the fault presents, and the result from automatically attempting to reproduce the fault within the fuzzing session.

**Findings report**

The Peach Platform generates a findings report that includes the following sections:

- Overall summary
- Faults
- Metrics

The summary is an executive summary that begins with a single status that can act as a dashboard indicator: "PASSED" or "FAILED". Additional information provides supporting detail: fuzzing configuration, number of issues and test cases, and details needed to reproduce the fuzzing session.

The Faults section enumerates the issues identified in the fuzzing job, including the following: vulnerabilities, unstable states, undesirable conditions, and crashes. The Peach Platform identifies whether the fault is reproducible, the risk associated with the fault, the number of times the fault occurred, and details about the monitor that detected the fault and the mutator that created the malformed data.

The metrics look at the test results and the test cases performed in the test session from different statistical perspectives. Each perspective offers a slightly different view that can be useful in evaluating the effectiveness of the fuzzing job.

The metrics in the fuzzing report include the following:

- The Mutators section focuses on faults produced by test cases that used a specific mutator
- The Elements section focuses on faults produced by test cases that used a specific data element
- The States section identifies the number of times the platform reached each defined state during the fuzzing job
- The Datasets section focuses on faults produced by test cases that used a specific dataset
- The Buckets section focuses on fault categories occurring in the fuzzing session, listing each category, the fault count in the category, risk associated with the fault, and other relevant data
Detailed findings of individual faults

The Peach Platform includes a file system logger that saves crash and fuzzing information from each test case. The log files contain information to debug and to reproduce faulting conditions. The quality of the debugging information saved when a fault occurs has a massive effect in prioritizing, tracking down, and resolving issues.

Information from a debugger is the most helpful, as it describes the state of the crashing process in detail. This information is available when testing occurs on a machine running a supported OS (Windows, Linux, or OS X) with a debugger attached via a Peach agent, and includes the following:

- Stack trace
- Register contents
- Memory contents
- Crash analysis

Unfortunately, this information isn’t always available, as in the case of testing an embedded system that has limited access to the hardware, and to the software interfaces. In this case, the embedded system is a black box, and attaching a debugger is not a monitoring option. In this situation, debugging and monitoring are external to the system. As such, external monitors (such as Ping and syslog) become more important in capturing as much meaningful information as is feasible from the communication channels that enter and leave the embedded system.

- Ping monitor can detect whether the system is responsive to a network request
- Syslog monitor watches system messages for relevant information
- Other monitors can watch for messages or actions that originate from the test subject.

When debuggers provide information for logging and tracing through a fault, the Peach Platform provides an initial risk analysis that details each issue so that you can prioritize the faults that occurred. The platform uses the risk analysis tools available for your operating system.

The monitoring system is also extensible via the Peach software development kit (SDK), allowing users to add custom fault detection, data collection, and automation modules as needed.

How the Peach Platform enables customization

The Peach Platform is a flexible tool that provides out-of-the-box fuzzing on Windows, Linux, and OS X operating systems while also exposing a robust API allowing for customization and extensibility. The Peach Platform offers users the limitless extensibility they need to perform effective fuzzing on a range of test targets. Here are some examples:

- A new or proprietary protocol needs a fuzzing definition that includes data and state models
- A new device needs a custom monitor to provide automation, data collection, or fault detection to the test configuration
- An alternative logging system needs to integrate to the platform
New fuzzing definitions
The Peach Platform enables you to create custom Peach Pits to test custom or proprietary protocols, data formats, and systems. The same features used to implement the existing collection of protocol definitions are available for use in creating custom fuzzing definitions. Any extensions made to your Peach Platform become part of the local fuzzing platform and are available for use in custom fuzzing definitions. In addition, the Peach Platform documentation provides tutorials and many examples of the components used to build pre-defined fuzzing definitions.

New components
Using the .NET library and any .NET language, such as C#, you can develop custom components, such as a monitor, and then integrate them within fuzzing definitions.

Additionally, the SDK provides examples of custom Peach Agents in C, C++, and Python. Custom agents are useful when fuzzing embedded systems such as mobile devices. Such devices are not powerful enough to run a full Peach Agent, but can run programs in C, C++, or Python.

About Peach Fuzzer, LLC
Peach Fuzzer, LLC, is a leader in security fuzz testing, having provided Peach Professional and Peach Enterprise solutions to some of the largest organizations in the world.

Peach Professional solutions provide complete fuzzing experiences with the Peach Platform. Peach Professional boasts the Peach fuzzing engine with web-based UI for running tests and reporting test results. In addition, a choice of Pits (or Pit Packs) are included to target specific test scenarios—whether testing a file format or a network protocol.

Peach Enterprise solutions provide multiple concurrent instances of the Peach Platform and developer features to create and extend fuzzing definitions. Aimed at companies with larger testing needs—five or more concurrent instances of the fuzzer—these solutions continue where Peach Professional ends.

For consultants and OEMs, Peach Fuzzer, LLC, offers solutions geared to your needs as well.