OWASP Web Application Penetration Checklist

Version 1.1

Introduction

Penetration testing will never be an exact science where a complete list of all possible issues that should be tested can de defined. Indeed penetration is only an appropriate technique to test the security of web applications under certain circumstances. For information about what these circumstances are, and to learn how to build a testing framework and which testing techniques you should consider, we recommend reading the OWASP Testing Framework Part One (http://www.owasp.org). *Risk Management Guide for Information Technology Systems*, NIST 800-30 [1]describes vulnerabilities in operational, technical and management categories. Penetration testing alone does not really help identify operational and management vulnerabilities.

Many OWASP followers (especially financial services companies) however have asked OWASP to develop a checklist that they can use when they do undertake penetration testing to promote consistency among both internal testing teams and external vendors. As such this list has been developed to be used in several ways including;

- RFP Template
- Benchmarks
- Testing Checklist

This checklist provides issues that should be tested. It does not prescribe techniques that should be used.

Using this Checklist as an RFP Template

Some people expressed the need for a checklist from which they can request services from vendors and consulting companies to ensure consistency, and from which they can compare approaches and results on a level playing field. As such, this list can form the basis of a Request for Proposal for services to a vendor. In effect, you are asking the vendor to perform all of the services listed.

NB: If you or your company develops an RFP Template from this checklist, please share it with OWASP and the community. Send it to testing@owasp.org with the Subject [Testing Checklist RFP Template].

Using this Checklist as a Benchmark

Some people expressed the need for a checklist from which they can base their internal testing on and from which they can then use the result to develop metrics. Using the same checklist allows people to compare different applications and even different sources of development as "apples to apples". The OASIS WAS project (http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=was) will provide a set of vulnerability types that can be used as a classification scheme and therefore have been adopted into

[1] http://csrc.nist.gov/publications/nistpubs/index.html#sp800-30 – The revised version can be found at http://csrc.nist.gov/publications/drafts/SP800-30-RevA-draft.pdf

this checklist to help people sort data easier. For more information see the section on OASIS WAS below.

Using this Checklist as a Checklist

Of course many people will want to use this checklist as just that; a checklist or crib sheet. As such the list is written as a set of issues that need to be tested. It does not prescribe techniques that should be used (although examples are provided).

About the OWASP Testing Project (Parts One and Two)

The OWASP is currently working on a comprehensive Testing Framework. By the time you read this document Part One will be close to release and Part Two will be underway. Part One of the Testing Framework describes the Why, What, Where and When of testing the security of web applications and Part Two goes into technical details about how to look for specific issues using source code inspection and a penetration testing (for example exactly how to find SQL Injection flaws in code and through penetration testing). This check list is likely to become an Appendix to Part Two of the OWASP Testing framework along with similar check lists for source code review.

The OASIS WAS Standard

The issues identified in this check list are not ordered in a specific manner of importance or criticality. Several members of the OWASP Team are working on an XML standard to develop a way to consistently describe web application security issues at OASIS. The mission of OASIS is to drive the development, convergence, and adoption of structured information standards in the areas of e-business, web services, etc. For more information about OASIS you should view the website http://www.oasis-open.org.

We believe OASIS WAS will become a very important standard which will allow people to develop vulnerability management / risk management systems and processes on top of the data. As this work is taking place at an official standards body its independence of vendor bias or technology and the fact that its longevity can be guaranteed, makes it suitable to base your work on. Part of the OASIS WAS standard will be a set of vulnerability types. These are standard vulnerability issues that will have standard textual definitions that allow people to build consistent classification schemes / thesauruses. Using these vulnerability types people can create useful views into their vulnerability data.

The OASIS WAS XL standard is due to be published in August. The WAS Vulnerability Types are due to be published as a separate document in draft at the end of April. As such this list “may” change when the standard is ratified although this is unlikely.

As we believe the WAS vulnerability types will become an integral part of application vulnerability management in the future, it will be tightly coupled to all OWASP work such as this checklist and the OWASP Testing Framework.

About OWASP

OWASP is a volunteer organization that is dedicated to developing knowledge based documentation and reference implementations and software that can be used by system architects, developers and security professionals. Our work promotes and helps consumers build more secure web applications.

For more information about OWASP see the web site http://www.owasp.org

Feedback

To provide feedback on this checklist please send an email to testing@owasp.org with a subject [Pen Testing Checklist Feedback]. We welcome all comments and suggestions. If your suggestion is for a new issue, please detail the issue as you would like to see it in the checklist. If your suggestion is a correction or improvement, please send your comments and a suggested completed text for the change. As a volunteer group, the easier your changes are to make, the faster they can be incorporated into our revisions.

Penetration Testing Workflow

Clearly, by promoting a checklist we are promoting methodical and repeatable testing. Whilst it is beyond scope of this checklist to prescribe a penetration testing methodology (this will be covered in OWASP Testing Part Two), we have included a model testing workflow below. Below is a flow diagram that the tester may find useful when using the testing techniques described in this document. It is important to note that an infrastructure level penetration test should be performed prior to performing the application test. In some cases, the server operating system can be exploited and give the tester further leverage in exploiting the web application.

The flow diagram below is based around several steps:

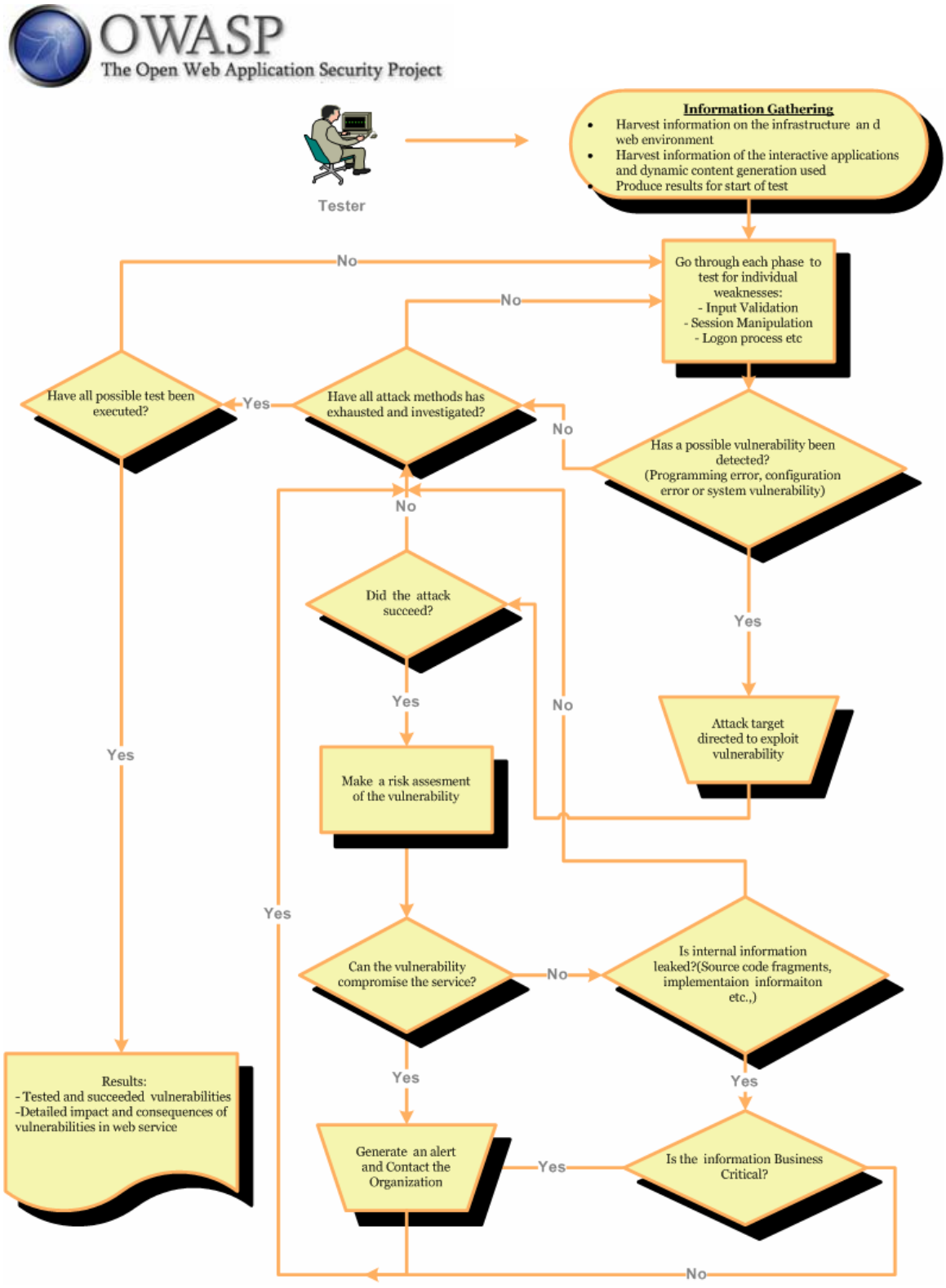
- The penetration test starts by gathering all possible information available regarding the infrastructure and applications involved. This stage is paramount as without a solid understanding of the underlying technology involved, sections may be missed during the testing phase

- The test should follow all the different phases described below

- Testers should attempt to exploit all discovered vulnerabilities. Even if the exploitation fails, the tester will have a better understanding of the vulnerability risk

- Any information returned by checking for vulnerabilities, for example, programming errors, source code retrieval or other internal information disclosure, should be used to re-assess the overall understanding of the application and how it performs

- If at any point during the testing a vulnerability is detected, which may lead to the successful compromise of the target or disclose business-critical information, the relevant contact for the company should be contacted immediately and made aware of the situation and risks involved.



Checklist

Category	Ref Number	Name	Objective	Notes
AppDOS	OWASP-AD-001	Application Flooding	Ensure that the application functions correctly when presented with large volumes of requests, transactions and / or network traffic.	Use various fuzzing tools to perform this test (e.g. SPIKE)
	OWASP-AD-002	Application Lockout	Ensure that the application does not allow an attacker to reset or lockout user's accounts.	

Category	Ref Number	Name	Objective	Notes
AccessControl	OWASP-AC-001	Parameter Analysis	Ensure that the application enforces its access control model by ensuring that any parameters available to an attacker would not afford additional service.	Typically this includes manipulation of form fields, URL query strings, client-side script values and cookies.
	OWASP-AC-002	Authorization	Ensure that resources that require authorization perform adequate authorization checks before being sent to a user.	
	OWASP-AC-003	Authorization Parameter Manipulation	Ensure that once valid user has logged in it is not possible to change the session ID's parameter to reflect another user account	i.e. accountnumber, policynumber,usernr etc.
	OWASP-AC-004	Authorized pages/functions	Check to see if its possible to access pages or functions which require logon but can be bypassed	
	OWASP-AC-005	Application Workflow	Ensure that where the application requires the user to perform actions in a specific sequence, the sequence is enforced.	

Category	Ref Number	Name	Objective	Notes
Authentication	OWASP-AUTHN-001	Authentication endpoint request should be HTTPS	Ensure that users are only asked to submit authentication credentials on pages that are served with SSL.	This ensures that the user knows who is asking for his / her credentials as well as where they are being sent.
	OWASP-AUTHN-002	Authentication bypass	Ensure that the authentication process can not be bypassed.	Typically this happens in conjunction with flaws like SQL Injection.
Authentication. User	OWASP-AUTHN-003	Credentials transport over an encrypted channel	Ensure that usernames and passwords are sent over an encrypted channel.	Typically this should be SSL.
	OWASP-AUTHN-004	Default Accounts	Check for default account names and passwords in use	
	OWASP-AUTHN-005	Username	Ensure that the username is not public (or "wallet") information such as email or SSN.	
	OWASP-AUTHN-006	Password Quality	Ensure that the password complexity makes guessing passwords difficult.	
	OWASP-AUTHN-007	Password Reset	Ensure that user must respond to a secret answer / secret question or other predetermined information before passwords can be reset.	Ensure that passwords are not sent to users in email.

Category	Ref Number	Name	Objective	Notes
	OWASP-AUTHN-008	Password Lockout	Ensure that the users account is locked out for a period of time when the incorrect password is entered more that a specific number of times (usually 5).	
	OWASP-AUTHN-009	Password Structure	Ensure that special meta characters cannot be used within the password	Can be useful when performing SQL injection
	OWASP-AUTHN-010	Blank Passwords	Ensure that passwords are not blank	
Authentication. SessionManagement	OWASP-AUTHSM-001	Session Token Length	Ensure that the session token is of adequate length to provide protection from guessing during an authenticated session.	
	OWASP-AUTHSM-002	Session Timeout	Ensure that the session tokens are only valid for a predetermined period after the last request by the user.	
	OWASP-AUTHSM-003	Session Reuse	Ensure that session tokens are changed when the user moves from an SSL protected resource to a non-SSL protected resource.	
	OWASP-AUTHSM-004	Session Deletion	Ensure that the session token is invalidated when the user logs out.	
	OWASP-AUTHSM-005	Session Token Format	Ensure that the session token is non-persistent and is never written to the browsers history or cache.	

Category	Ref Number	Name	Objective	Notes
Configuration. Management	OWASP-CM-001	HTTP Methods	Ensure that the web server does not support the ability to manipulate resources from the Internet (e.g. PUT and DELETE)	
	OWASP-CM-002	Virtually Hosted Sites	Try and determine if site is virtually hosted.	If there are further sites, they could be vulnerable and lead to the compromise of the base server
	OWASP-CM-003	Known Vulnerabilities / Security Patches	Ensure that known vulnerabilities which vendors have patched are not present.	
	OWASP-CM-004	Back-up Files	Ensure that no backup files of source code are accessible on the publicly accessible part of the application.	
	OWASP-CM-004	Web Server Configuration	Ensure that common configuration issues such as directory listings and sample files have been addressed	
	OWASP-CM-005	Web Server Components	Ensure that web server components like Front Page Server Extensions or Apache modules do not introduce any security vulnerabilities	
	OWASP-CM-006	Common Paths	Check for existence of common directories within the application root	/backup & /admin may contain information

Category	Ref Number	Name	Objective	Notes
	OWASP-CM-007	Language/Application defaults	I.e. J2EE environmental quirks e.g Availability of snoop.jsp /*Spy.jsp and loaded modules	
Configuration. Management Infrastructure	OWASP-CM-008	Infrastructure Admin Interfaces	Ensure that administrative interfaces to infrastructure such as web servers and application servers are not accessible to the Internet.	
Configuration. Management. Application	OWASP-CM-009	Application Admin Interfaces	Ensure that administrative interfaces to the applications are not accessible to the Internet.	

Category	Ref Number	Name	Objective	Notes
Error Handling	OWASP-EH-001	Application Error Messages	Ensure that the application does not present application error messages to an attacker that could be used in an attack.	This typically occurs when applications return verbose error messages such as stack traces or database errors.
	OWASP-EH-002	User Error Messages	Ensure that the application does not present user error messages to an attacker that could be used in an attack.	This typically occurs when applications return error messages such as “User does not exist” or “User Correct, Password Incorrect”

Category	Ref Number	Name	Objective	Notes
DataProtection	OWASP-DP-001	Sensitive Data in HTML	Ensure that there is no sensitive data in the HTML (cached in the browser history) that could lead an attacker to mount a focused attack.	This typically occurs when developers leave information in html comment or the application renders names and addresses in HTML.
	OWASP-DP-002	Data Storage	Ensure where required, data is protected to protect its confidentiality and integrity.	
DataProtection. Transport	OWASP-DP-003	SSL Version	Ensure that SSL versions supported do not have cryptographic weaknesses.	Typically this means supporting SSL 3 and TLS 1.0 only.
	OWASP-DP-004	SSL Key Exchange Methods	Ensure that the web server does not allow anonymous key exchange methods.	Typically ADH Anonymous Diffie-Hellman.
	OWASP-DP-005	SSL Algorithms	Ensure that weak algorithms are not available.	Typically algorithms such as RC2 and DES.
	OWASP-DP-006	SSL Key Lengths	Ensure the web site uses an appropriate length key.	Most web sites should enforce 128 bit encryption.
	OWASP-DP-007	Digital Certificate Validity	Ensure the application uses valid digital certificates.	Ensure that the digital certificate is valid, that is to say its signature, host, date etc are all valid.

Category	Ref Number	Name	Objective	Notes
InputValidation	OWASP-IV-001	Script Injection	Ensure that any part of the application that allows input does not process scripts as part of the input.	Classic case of Cross Site Scripting but includes other scripting as well.
InputValidation.SQL	OWASP-IV-002	SQL Injection	Ensure the application will not process SQL commands from the user.	
InputValidation.OS	OWASP-IV-003	OS Command Injection	Ensure the applications will not process operating system commands from the user.	This typically includes issues such as path traversal, spawning command shells and OS functions.
InputValidation.LDAP	OWASP-IV-004	LDAP Injection	Ensure the application will not process LDAP commands form the user.	
InputValidation.XSS	OWASP-IV-005	Cross Site Scripting	Ensure that the application will not store or reflect malicious script code.	

Category	Ref Number	Name	Objective	Notes
BufferOverflow	OWASP-BO-001	Overflows	Ensure that the application is not susceptible to any buffer overflows.	Fuzzing tools help with testing all components of an application for this issue.
	OWASP-BO-002	Heap Overflows	Ensure that the application is not susceptible to any heap overflows.	
	OWASP-BO-003	Stack Overflows	Ensure that the application is not susceptible to any stack overflows.	
	OWASP-BO-004	Format Strings	Ensure that the application is not susceptible to any format string overflows.	

Appendix A - The OASIS WAS Vulnerability Types

AppDOS
Flaws which may would allow an attacker to completely or partially prevent users from using an application properly.
AppDOS.Flood
Used for application denial of service problems that involve saturating a limited resource shared by all users of the application, such as disk space, CPU, network bandwidth, database connections, or memory.
AppDOS.Lockout
Used for application denial of service problems that involve using up a resource or limit allocated to users, such as failed logon attempts, messages, or transactions.
AccessControl
Problems which may allow users to access assets or functions they are not authorized for. Frequently, there is no access control mechanism where there should be. A proper access control mechanism should enforce the principles of a reference monitor: tamperproof, and analyzable.
Authentication
Used for problems related to determining the identity of individuals or entities and authenticating that identity.
Authentication.User
Used for issues related to identification and authentication of people who are intended to use an application. Problems with usernames, passwords, tokens, smartcards, biometrics, and other credentials are examples.
Authentication.UserManagement
Used for problems related to managing a set of users, especially the security relevant information such as roles, privileges, authorizations, groups, social security numbers, credit card numbers, and other sensitive information. Also problems with creating new users, registration, granting rights, and terminating access.
Authentication.Entity
Used for problems with authenticating automated systems, such as web services, databases, directories, and others. Examples include secure credential storage, securing transport, changing credentials, and terminating access.
Authentication.SessionManagement
Used for problems with issuing, using, protecting, changing, and terminating session identifiers of all kinds. Session identifiers stand in the place of authentication credentials yet are frequently not protected as carefully.
BufferOverflow
Flaws which may allow an attacker to use format strings to overwrite locations in memory, allowing data to be changed, program control to be altered, or the program to crash.
BufferOverflow.Heap
Flaws which may allow an attacker to overflow memory that is dynamically allocated by the application.

BufferOverflow.Stack
Flaws which may allow an attacker to write data into the stack, causing the program to crash or transfer control.
BufferOverflow.Format
Flaws which may allow an attacker to use format strings to overwrite locations in memory, allowing data to be changed, program control to be altered, or the program to crash.
Concurrency
Used for errors in multithreaded environments that allows data to be shared or corrupted. Examples include variables that are shared between threads and cause time-of-check-time-of-use (TOCTOU) problems, broken singleton patterns, and poor cache design.
ConfigurationManagement
Used to describe problems in the configuration of an application or application environment.
ConfigurationManagement.Administration
Used for problems in the application's mechanisms that enable remote administration, such as user management, credential management, database management, and other configuration options.
ConfigurationManagement.Application
Used to describe problems in the application's configuration, such as mis-configured security mechanisms, default programs, unused code, and unnecessarily enabled features.
ConfigurationManagement.Infrastrure
Used for problems with the configuration of the application's infrastructure, such as the web and application servers, filters, and external security mechanisms.
Cryptography
Used for problems related to encryption, decryption, signing, and verification.
Cryptography.Algorithm
Used for cryptographic algorithm selection, implementation, and analysis problems.
Cryptography.KeyManagement
Used for issues with certificate storage, tokens, revocation, certificates, key stores, issuing keys, and other key issues
DataProtection
Used for issues related to inappropriate disclosure of data.
DataProtection.Storage
Used for problems storing data securely, including storage of credentials, keys, and other sensitive information. Mistakes related to cryptographic mechanisms include poor sources of randomness, bad choice of algorithm, and poor implementation.
DataProtection.Transport
Used for problems related to secure transfer of information. Frequently, this will refer to problems with SSL or TLS configuration, but could include other protocols with security features.
ErrorHandling
Used for problems in handling errors, including printing stack traces to the screen, fail open security mechanisms, allowing errors to affect the operation of the entire application, and revealing too much information about a failure.

InputValidation
Used for issues related to failure to validate un-trusted input before it is relied on by an application.
InputValidation.User
Used for input validation problems where the input comes from a human user, such as HTTP request parameters, command line input, or input events from an application's GUI.
InputValidation.Network
Used for input validation problems where the input comes from a network protocol, such as HTTP headers, sequence numbers, or other protocol fields.
InputValidation.File
Used for input validation problems where the input comes from a file, such as a properties file, batch data file, flat-file databases, or other file based data.
Injection
Problems which may allow an attacker to bury commands into data and have them interpreted by some system that the data reaches.
Injection.SQL
Flaws which may allow an attacker to inject special characters and commands into a SQL database and modify the intended query. The attack might attempt to change the meaning of the query, or might attempt to chain additional commands.
Injection.HTML
Flaws which may allow an attacker to inject HTML into an application and modify the appearance of HTML generated by that application. For example, an attacker might inject an unwanted IMG tag into a guest book, and offend other users.
Injection.OSCommand
Flaws which may allow an attacker to inject special characters and commands into the operating system command shell and modify the intended command. The attack might attempt to modify how a program is invoked, or might attempt to chain additional commands.
Injection.LDAP
Flaws which may allow an attacker to inject special characters and search terms into an LDAP server and modify the intended query.
Injection.XSS
Flaws which may allow an attacker to send and execute malicious scripts through a web application. Stored XSS attacks store the script in the web application. Reflected XSS attacks are bounced off a web application in real time and require a user to be tricked into sending the request containing the attack.
Monitoring
Used for issues related to monitoring the security posture of a web application.
Monitoring.Logging
Used for issues concerning the proper logging of events, including what should be logged, how it should be logged, how logs should be reviewed, and other issues related to accountability.
Monitoring.Detection
Used for issues related to the detection of attacks on an application, how attacks should be handled, what information should be gathered, and who should be notified.