| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/SecurityPermission.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/security/Security.html)   [**NEXT CLASS**](http://docs.google.com/java/security/Signature.html) | [**FRAMES**](http://docs.google.com/index.html?java/security/SecurityPermission.html)    [**NO FRAMES**](http://docs.google.com/SecurityPermission.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#tyjcwt) | DETAIL: FIELD | [CONSTR](#4d34og8) | METHOD |

## **java.security**

Class SecurityPermission

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 [java.security.Permission](http://docs.google.com/java/security/Permission.html)  
 [java.security.BasicPermission](http://docs.google.com/java/security/BasicPermission.html)  
 **java.security.SecurityPermission**

**All Implemented Interfaces:** [Serializable](http://docs.google.com/java/io/Serializable.html), [Guard](http://docs.google.com/java/security/Guard.html)

public final class **SecurityPermission**extends [BasicPermission](http://docs.google.com/java/security/BasicPermission.html)

This class is for security permissions. A SecurityPermission contains a name (also referred to as a "target name") but no actions list; you either have the named permission or you don't.

The target name is the name of a security configuration parameter (see below). Currently the SecurityPermission object is used to guard access to the Policy, Security, Provider, Signer, and Identity objects.

The following table lists all the possible SecurityPermission target names, and for each provides a description of what the permission allows and a discussion of the risks of granting code the permission.

| Permission Target Name | What the Permission Allows | Risks of Allowing this Permission |
| --- | --- | --- |
| createAccessControlContext | Creation of an AccessControlContext | This allows someone to instantiate an AccessControlContext with a DomainCombiner. Since DomainCombiners are given a reference to the ProtectionDomains currently on the stack, this could potentially lead to a privacy leak if the DomainCombiner is malicious. |
| getDomainCombiner | Retrieval of an AccessControlContext's DomainCombiner | This allows someone to retrieve an AccessControlContext's DomainCombiner. Since DomainCombiners may contain sensitive information, this could potentially lead to a privacy leak. |
| getPolicy | Retrieval of the system-wide security policy (specifically, of the currently-installed Policy object) | This allows someone to query the policy via the getPermissions call, which discloses which permissions would be granted to a given CodeSource. While revealing the policy does not compromise the security of the system, it does provide malicious code with additional information which it may use to better aim an attack. It is wise not to divulge more information than necessary. |
| setPolicy | Setting of the system-wide security policy (specifically, the Policy object) | Granting this permission is extremely dangerous, as malicious code may grant itself all the necessary permissions it needs to successfully mount an attack on the system. |
| createPolicy.{policy type} | Getting an instance of a Policy implementation from a provider | Granting this permission enables code to obtain a Policy object. Malicious code may query the Policy object to determine what permissions have been granted to code other than itself. |
| getProperty.{key} | Retrieval of the security property with the specified key | Depending on the particular key for which access has been granted, the code may have access to the list of security providers, as well as the location of the system-wide and user security policies. while revealing this information does not compromise the security of the system, it does provide malicious code with additional information which it may use to better aim an attack. |
| setProperty.{key} | Setting of the security property with the specified key | This could include setting a security provider or defining the location of the the system-wide security policy. Malicious code that has permission to set a new security provider may set a rogue provider that steals confidential information such as cryptographic private keys. In addition, malicious code with permission to set the location of the system-wide security policy may point it to a security policy that grants the attacker all the necessary permissions it requires to successfully mount an attack on the system. |
| insertProvider.{provider name} | Addition of a new provider, with the specified name | This would allow somebody to introduce a possibly malicious provider (e.g., one that discloses the private keys passed to it) as the highest-priority provider. This would be possible because the Security object (which manages the installed providers) currently does not check the integrity or authenticity of a provider before attaching it. |
| removeProvider.{provider name} | Removal of the specified provider | This may change the behavior or disable execution of other parts of the program. If a provider subsequently requested by the program has been removed, execution may fail. Also, if the removed provider is not explicitly requested by the rest of the program, but it would normally be the provider chosen when a cryptography service is requested (due to its previous order in the list of providers), a different provider will be chosen instead, or no suitable provider will be found, thereby resulting in program failure. |
| setSystemScope | Setting of the system identity scope | This would allow an attacker to configure the system identity scope with certificates that should not be trusted, thereby granting applet or application code signed with those certificates privileges that would have been denied by the system's original identity scope |
| setIdentityPublicKey | Setting of the public key for an Identity | If the identity is marked as "trusted", this allows an attacker to introduce a different public key (e.g., its own) that is not trusted by the system's identity scope, thereby granting applet or application code signed with that public key privileges that would have been denied otherwise. |
| setIdentityInfo | Setting of a general information string for an Identity | This allows attackers to set the general description for an identity. This may trick applications into using a different identity than intended or may prevent applications from finding a particular identity. |
| addIdentityCertificate | Addition of a certificate for an Identity | This allows attackers to set a certificate for an identity's public key. This is dangerous because it affects the trust relationship across the system. This public key suddenly becomes trusted to a wider audience than it otherwise would be. |
| removeIdentityCertificate | Removal of a certificate for an Identity | This allows attackers to remove a certificate for an identity's public key. This is dangerous because it affects the trust relationship across the system. This public key suddenly becomes considered less trustworthy than it otherwise would be. |
| printIdentity | Viewing the name of a principal and optionally the scope in which it is used, and whether or not it is considered "trusted" in that scope | The scope that is printed out may be a filename, in which case it may convey local system information. For example, here's a sample printout of an identity named "carol", who is marked not trusted in the user's identity database:  carol[/home/luehe/identitydb.obj][not trusted] |
| clearProviderProperties.{provider name} | "Clearing" of a Provider so that it no longer contains the properties used to look up services implemented by the provider | This disables the lookup of services implemented by the provider. This may thus change the behavior or disable execution of other parts of the program that would normally utilize the Provider, as described under the "removeProvider.{provider name}" permission. |
| putProviderProperty.{provider name} | Setting of properties for the specified Provider | The provider properties each specify the name and location of a particular service implemented by the provider. By granting this permission, you let code replace the service specification with another one, thereby specifying a different implementation. |
| removeProviderProperty.{provider name} | Removal of properties from the specified Provider | This disables the lookup of services implemented by the provider. They are no longer accessible due to removal of the properties specifying their names and locations. This may change the behavior or disable execution of other parts of the program that would normally utilize the Provider, as described under the "removeProvider.{provider name}" permission. |
| getSignerPrivateKey | Retrieval of a Signer's private key | It is very dangerous to allow access to a private key; private keys are supposed to be kept secret. Otherwise, code can use the private key to sign various files and claim the signature came from the Signer. |
| setSignerKeyPair | Setting of the key pair (public key and private key) for a Signer | This would allow an attacker to replace somebody else's (the "target's") keypair with a possibly weaker keypair (e.g., a keypair of a smaller keysize). This also would allow the attacker to listen in on encrypted communication between the target and its peers. The target's peers might wrap an encryption session key under the target's "new" public key, which would allow the attacker (who possesses the corresponding private key) to unwrap the session key and decipher the communication data encrypted under that session key. |

**See Also:**[BasicPermission](http://docs.google.com/java/security/BasicPermission.html), [Permission](http://docs.google.com/java/security/Permission.html), [Permissions](http://docs.google.com/java/security/Permissions.html), [PermissionCollection](http://docs.google.com/java/security/PermissionCollection.html), [SecurityManager](http://docs.google.com/java/lang/SecurityManager.html), [Serialized Form](http://docs.google.com/serialized-form.html#java.security.SecurityPermission)

| **Constructor Summary** | |
| --- | --- |
| [**SecurityPermission**](http://docs.google.com/java/security/SecurityPermission.html#SecurityPermission(java.lang.String))([String](http://docs.google.com/java/lang/String.html) name)            Creates a new SecurityPermission with the specified name. |
| [**SecurityPermission**](http://docs.google.com/java/security/SecurityPermission.html#SecurityPermission(java.lang.String,%20java.lang.String))([String](http://docs.google.com/java/lang/String.html) name, [String](http://docs.google.com/java/lang/String.html) actions)            Creates a new SecurityPermission object with the specified name. |

| **Method Summary** | |
| --- | --- |

| **Methods inherited from class java.security.**[**BasicPermission**](http://docs.google.com/java/security/BasicPermission.html) |
| --- |
| [equals](http://docs.google.com/java/security/BasicPermission.html#equals(java.lang.Object)), [getActions](http://docs.google.com/java/security/BasicPermission.html#getActions()), [hashCode](http://docs.google.com/java/security/BasicPermission.html#hashCode()), [implies](http://docs.google.com/java/security/BasicPermission.html#implies(java.security.Permission)), [newPermissionCollection](http://docs.google.com/java/security/BasicPermission.html#newPermissionCollection()) |

| **Methods inherited from class java.security.**[**Permission**](http://docs.google.com/java/security/Permission.html) |
| --- |
| [checkGuard](http://docs.google.com/java/security/Permission.html#checkGuard(java.lang.Object)), [getName](http://docs.google.com/java/security/Permission.html#getName()), [toString](http://docs.google.com/java/security/Permission.html#toString()) |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [clone](http://docs.google.com/java/lang/Object.html#clone()), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

| **Constructor Detail** |
| --- |

### SecurityPermission

public **SecurityPermission**([String](http://docs.google.com/java/lang/String.html) name)

Creates a new SecurityPermission with the specified name. The name is the symbolic name of the SecurityPermission. An asterisk may appear at the end of the name, following a ".", or by itself, to signify a wildcard match.

**Parameters:**name - the name of the SecurityPermission **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if name is null. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if name is empty.

### SecurityPermission

public **SecurityPermission**([String](http://docs.google.com/java/lang/String.html) name,  
 [String](http://docs.google.com/java/lang/String.html) actions)

Creates a new SecurityPermission object with the specified name. The name is the symbolic name of the SecurityPermission, and the actions String is currently unused and should be null.

**Parameters:**name - the name of the SecurityPermissionactions - should be null. **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if name is null. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if name is empty.

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/SecurityPermission.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/security/Security.html)   [**NEXT CLASS**](http://docs.google.com/java/security/Signature.html) | [**FRAMES**](http://docs.google.com/index.html?java/security/SecurityPermission.html)    [**NO FRAMES**](http://docs.google.com/SecurityPermission.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#tyjcwt) | DETAIL: FIELD | [CONSTR](#4d34og8) | METHOD |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).