Security Extension Specification

X Consortium Standard

David P. Wiggins, X Consortium

Security Extension Specification: X Consortium Standard

by David P. Wiggins

X Version 11, Release 7.7

Version 7.1

Publication date November 15, 1996

Copyright © 1996 X Consortium

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE X CONSORTIUM BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Except as contained in this notice, the name of the X Consortium shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization from the X Consortium.

X Window System is a trademark of The OpenGroup.

Table of Contents

Chapter 1. Introduction

The Security extension contains new protocol needed to provide enhanced X server security. The Security extension should not be exposed to untrusted clients (defined below).

Chapter 2. Requests

SecurityQueryVersion

This request returns the major and minor version numbers of this extension.

SecurityQueryVersion

client-major-version	CARD16
client-minor-version	CARD16
=>	
server-major-version	CARD16
server-minor-version	CARD16

The client-major-version and client-minor-version numbers indicate what version of the protocol the client wants the server to implement. The server-major-version and the server-minor-version numbers returned indicate the protocol this extension actually supports. This might not equal the version sent by the client. An implementation can (but need not) support more than one version simultaneously. The server-major-version and server-minor-version allow the creation of future revisions of the Security protocol that may be necessary. In general, the major version would increment for incompatible changes, and the minor version would increment for small, upward-compatible changes. Servers that support the protocol defined in this document will return a server-major-version of one (1), and a server-minor-version of zero (0).

Clients using the Security extension must issue a SecurityQueryVersion request before any other Security request in order to negotiate a compatible protocol version; otherwise, the client will get undefined behavior (Security may or may not work).

SecurityGenerateAuthorization

This request causes the server to create and return a new authorization with specific characteristics. Clients can subsequently connect using the new authorization and will inherit some of the characteristics of the authorization.

SecurityGenerateAuthorization

authorization-protocol-name	STRING8
authorization-protocol-data	STRING8
value-mask	BITMASK
value-list	LISTofVALUE
=>	
authorization-id	AUTHID
authorization-data-return	STRING8

Errors: `AuthorizationProtocol`, `Value`, `Alloc` authorization-protocol-name is the name of the authorization method for which the server should generate a new authorization that subsequent clients can use to connect to the server. If the authorization-protocol-name is not one that the server supports, or if authorization-protocol-data does not make sense for the given authorization-protocol-name, an AuthorizationProtocol error results.

authorization-protocol-data is authorization-method specific data that can be used in some way to generate the authorization.

Note

In this version of the extension, the only authorization method required to be supported is "MIT-MAGIC-COOKIE-1" with any amount of authorization-protocol-data (including none). The server may use the authorization-protocol-data as an additional source of randomness used to generate the authorization. Other authorization methods can supply their own interpretation of authorization-protocol-data.

The value-mask and value-list specify attributes of the authorization that are to be explicitly initialized. The possible values are:

Attribute	**Type**	**Default**
timeout	CARD32	60
group	XID or None	None
trust-level	{SecurityClientTrusted,	
	SecurityClientUntrusted}	SecurityClientUntrusted
event-mask	SecurityAuthorizationRevoked,	
	or None	None

timeout is the timeout period in seconds for this authorization. A timeout value of zero means this authorization will never expire. For non-zero timeout values, when timeout seconds have elapsed since the last time that the authorization entered the state of having no connections authorized by it, and if no new connections used the authorization during that time, the authorization is automatically purged. (Note that when an authorization is created, it enters the state of having no connections authorized by it.) Subsequent connection attempts using that authorization will fail. This is to facilitate "fire and forget" launching of applications.

group is an application group ID as defined by the Application Group extension, or None. Any other values will cause a Value error. When a group is destroyed, all authorizations specifying that group are revoked as described under the SecurityRevokeAuthorization request. The Application Group extension attaches additional semantics to the group.

trust-level tells whether clients using the authorization are trusted or untrusted. If trust-level is not one of the constants SecurityClientTrusted or SecurityClientUntrusted, a Value error results.

event-mask defines which events the client is interested in for this authorization. When the authorization expires or is revoked if event-mask contains SecurityAuthorizationRevoked a SecurityAuthorizationRevoked event is reported to the client.

The SecurityAuthorizationRevoked event contains the following field:

Field	Type
authorization-id	AUTHID

where authorization-id is the identification of the authorization that was revoked.

If an invalid value-mask is specified, a Value error occurs.

The returned authorization-id is a non-zero value that uniquely identifies this authorization for use in other requests. The value space for type AUTHID is not required to be disjoint from values spaces of other core X types, e.g. resource ids, atoms, visual ids, and keysyms. Thus, a given numeric value might be both a valid AUTHID and a valid atom, for example.

authorization-data-return is the data that a client should use in some authorization-method-specific way to make a connection with this authorization. For "MIT-MAGIC-COOKIE-1," authorization-data-return should be sent as the authorization-protocol-data in the connection setup message. It is not required that other authorization methods use authorization-data-return this way.

SecurityRevokeAuthorization

This request deletes an authorization created by SecurityGenerateAuthorization.

SecurityRevokeAuthorization

authorization-id	AUTHID

Errors: `Authorization`

If authorization-id does not name a valid authorization, an Authorization error occurs. Otherwise, this request kills all clients currently connected using the authorization specified by authorization-id. The authorization is deleted from the server's database, so future attempts by clients to connect with this authorization will fail.

Chapter 3. Changes to Core Requests

A server supporting this extension modifies the handling of some core requests in the following ways.

Resource ID Usage

If an untrusted client makes a request that specifies a resource ID that is not owned by another untrusted client, a protocol error is sent to the requesting client indicating that the specified resource does not exist. The following exceptions apply. An untrusted client can:

1. use the QueryTree, GetGeometry, and TranslateCoordinates requests without restriction.
2. use colormap IDs that are returned in the default-colormap field of its connection setup information in any colormap requests.
3. specify a root window as:
    a. the drawable field of CreatePixmap, CreateGC, and QueryBestSize.
    b. the parent field of CreateWindow.
    c. the window field of CreateColormap, ListProperties, and GetWindowAttributes.
    d. the grab-window or confine-to fields of GrabPointer.
    e. the grab-window field of UngrabButton.
    f. the destination of SendEvent, but only if all of the following are true. (These conditions cover all the events that the ICCCM specifies with a root window destination.)
        i. The propogate field of SendEvent is False.
        ii. The event-mask field of SendEvent is ColormapChange, StructureNotify, or the logical OR of SubstructureRedirect with SubstructureNotify.
        iii. The event type being sent is UnmapNotify, ConfigureRequest, or ClientMessage.
    g. the window field of ChangeWindowAttributes, but only if the value-mask contains only event-mask and the corresponding value is StructureNotify, PropertyChange, or the logical OR of both.

Note

ISSUE: are root window exceptions needed for these? WarpPointer, ReparentWindow (parent), CirculateWindow, QueryPointer (emacs does this), GetMotionEvents.

Extension Security

This extension introduces the notion of secure and insecure extensions. A secure extension is believed to be safe to use by untrusted clients; that is, there are no significant security concerns known that an untrusted client could use to destroy, modify, or steal data of trusted clients. This belief may be founded on a careful analysis of the extension protocol, its implementation, and measures taken to "harden" the extension to close security weaknesses. All extensions not considered secure are called insecure. The implementation details of how an extension is identified as secure or insecure are beyond the scope of this specification.

`ListExtensions` will only return names of secure extensions to untrusted clients.

If an untrusted client uses `QueryExtension` on an insecure extension that the server supports, the reply will have the present field set to False and the major-opcode field set to zero to indicate that the extension is not supported.

If an untrusted client successfully guesses the major opcode of an insecure extension, attempts by it to execute requests with that major opcode will fail with a Request error.

Keyboard Security

The protocol interpretation changes in this section are intended to prevent untrusted applications from stealing keyboard input that was meant for trusted clients and to prevent them from interfering with the use of the keyboard.

The behavior of some keyboard-related requests and events is modified when the client is untrusted depending on certain server state at the time of request execution or event generation. Specifically, if a hypothetical keyboard event were generated given the current input focus, pointer position, keyboard grab state, and window event selections, and if that keyboard event would not be delivered to any untrusted client, the following changes apply:

1. The bit vector representing the up/down state of the keys returned by `QueryKeymap` and `KeymapNotify` is all zeroes.

2. GrabKeyboard returns a status of AlreadyGrabbed.

3. `SetInputFocus` does nothing. Note that this means the Globally Active Input and WM_TAKE_FOCUS mechanisms specified in the ICCCM will not work with untrusted clients.

4. Passive grabs established by GrabKey that would otherwise have activated do not activate.

If an untrusted client attempts to use any of the following requests, the only effect is that the client receives an Access error: SetModifierMapping, ChangeKeyboardMapping, ChangeKeyboardControl.

If an InputOnly window owned by an untrusted client has a parent owned by a trusted client, all attempts to map the window will be ignored. This includes mapping attempts resulting from MapWindow, MapSubwindows, ReparentWindow, and save-set processing.

However, if the parent of an InputOnly window owned by an untrusted client is the root window, attempts to map that window will be performed as expected. This is in line with the root window exceptions above.

Image Security

It should be impossible for an untrusted client to retrieve the image contents of a trusted window unless a trusted client takes action to allow this. We introduce the following defenses in support of this requirement.

The restrictions on resource ID usage listed above prevent untrusted clients from using GetImage directly on windows not belonging to trusted clients.

If an untrusted client tries to set the background-pixmap attribute of an untrusted window to None, the server will instead use a server-dependent background which must be different than None.

The X protocol description of `GetImage` states that the returned contents of regions of a window obscured by noninferior windows are undefined if the window has no backing store. Some implementations return the contents of the obscuring windows in these regions. When an untrusted client uses `GetImage`, this behavior is forbidden; the server must fill the obscured regions in the returned image with a server-dependent pattern.

If an untrusted window has trusted inferiors, their contents are vulnerable to theft via `GetImage` on the untrusted parent, as well as being vulnerable to destruction via drawing with subwindow-mode IncludeInferiors on the untrusted parent. An untrusted window having trusted inferiors can only occur at the request of a trusted client. It is expected to be an unusual configuration.

Property Security

Unlike the other security provisions described in this document, security for property access is not amenable to a fixed policy because properties are used for inter-client communication in diverse ways and may contain data of varying degrees of sensitivity. Therefore, we only list the possible restrictions the server may decide to impose on use of properties on trusted windows by untrusted clients. How the server chooses which restrictions from this list to apply to a particular property access is implementation dependent [1].

The X Protocol property requests are `ChangeProperty`, `GetProperty`, `DeleteProperty`, `RotateProperties`, and `ListProperties`. For these requests, the server can allow the request to execute normally (as if it had been issued by a trusted client), ignore the request completely (as if it were a NoOperation), or ignore the request except to send an Atom error to the client. Ignoring a `ListProperties` request means replying that the window has no properties. `ListProperties` may also reply with a subset of the existing properties if the server is doing property hiding; see below.

---

[1] In the X Consortium server implementation, property access is controlled by a configuration file; see the -sp option in the Xserver(1) manual page.

An ignored `GetProperty` request may reply that the property does not exist, or that it exists but contains no data.

The server may decide to hide certain properties on certain windows from untrusted clients [2]. If a property is to be hidden, it must be done consistently to avoid confusing clients. This means that for untrusted clients:

- That property should not be returned by `ListProperties`.
- `PropertyNotify` events should not be sent for that property.
- `GetProperty` on that property should reply that the property does not exist (the return type is None, the format and bytes-after are zero, and the value is empty).

For a property that the server is protecting but not hiding, consistency must also be maintained:

- That property should be returned by `ListProperties`.
- `PropertyNotify` events should be sent for that property.
- `GetProperty` on that property should reply that the property exists (if it really does) but the value is empty (return type and format are their real values, and the "length of value" field in the reply is zero).

Miscellaneous Security

If an untrusted client attempts to use `ChangeHosts`, `ListHosts`, or `SetAccessControl`, the only effect is that the client receives an Access error.

If an untrusted client attempts to use `ConvertSelection` on a selection with a trusted selection owner window, the server generates a SelectionNotify event to the requestor with property None.

---

[2] The X Consortium server implementation does not currently provide a way to hide properties.

Chapter 4. New Authorization Method

This extension includes a new authorization method named "XC-QUERY-SECURITY-1". Its purpose is to allow an external agent such as the X firewall proxy to probe an X server to determine whether that server meets certain security criteria without requiring the agent to have its own authorization for that server. The agent may use the returned information to make a decision. For example, the X firewall proxy may choose not to forward client connections to servers that do not meet the criteria.

To use this authorization method, the client (or proxy) sends "XC-QUERY-SECURITY-1" as the authorization-protocol-name in the initial connection setup message. The authorization-protocol-data may be empty or may contain additional security criteria desribed below. If the success field of the server's reply is Authenticate, the server supports the security extension, and the server meets all specified additional security criteria. In this case, the client should resend the initial connection setup message substituting the authorization protocol name and data that should be used to authorize the connection. If the success field of the server's reply is anything other than Authenticate, either the server does not support the security extension, does not meet (or cannot determine if it meets) all of the additional security criteria, or chooses for internal reasons not to answer with Authenticate. In this case, the client should close the connection.

If the authorization-protocol-data sent with "XC-QUERY-SECURITY-1" is not empty, it specifies additional security criteria for the server to check, as follows.

`authorization-protocol-data`

policy-mask	BITMASK
policies	LISTofSECURITYPOLICY

The policy-mask field is any logical-OR combination of the constants Extensions and SitePolicies. For each bit set in policy-mask, there is a SECURITYPOLICY element in policies. The nth element in policies corresponds to the nth 1-bit in policy-mask, counting upward from bit 0.

`SECURITYPOLICY`

policy-type	{Disallow, Permit}
names	LISTofSTR

For a SECURITYPOLICY corresponding to policy-mask Extensions, if policy-type is Disallow the server is required to consider as insecure all extensions given in names. No policy is specified for extensions not listed in names. If policy-type is Permit the server may consider only those extensions given in names to be secure; all other extensions must be treated as insecure. If these constraints are not met, the server should not return Authenticate in the success field of the reply. Servers can but need not dynamically configure themselves in response to an Extensions SECURITYPOLICY; a conforming server might simply compare the policy with a compiled-in table of extensions and their security status.

For a SECURITYPOLICY corresponding to policy-mask SitePolicies, policy-type Disallow means the server must not have been configured with any of the site policies given in names. Policy-type Permit means the server must have been configured with at least one of the site policies given in names. If these constraints are not met, the server should not return Authenticate in the success field of the reply.

SitePolicies provide a way to express new forms of security-relevant information that could not be anticipated at the time of this writing. For example, suppose the server is found to have a critical security defect. When a fix is developed, a site policy string could be associated with the fix. Servers with the fix would advertise that site policy, and the X firewall proxy would specify that site policy in a SECURITYPOLICY with policy-type Permit.

Chapter 5. Encoding

Please refer to the X11 Protocol Encoding document as this section uses syntactic conventions and data types established there.

The name of this extension is "SECURITY".

Types

AUTHID: CARD32

Request Encoding

SecurityQueryVersion

```
1      CARD8        major-opcode
1      0            minor-opcode
2      2            request length
2      CARD16       client-major-version
2      CARD16       client-minor-version
=>
1      1    Reply
1           unused
2      CARD16    sequence number
4      0    reply length
2      CARD16    server-major-version
2      CARD16    server-minor-version
20                   unused
```

`SecurityRevokeAuthorization`

```
1      CARD8        major-opcode
1      2            minor-opcode
2      2            request length
4      AUTHID       authorization-id
```

SecurityGenerateAuthorization

```
1      CARD8              major-opcode
1      1                  minor-opcode
2      3 + (m+n+3)/4 + s  request length
2      CARD16             m, number of bytes in authorization protocol name
2      CARD16             n, number of bytes in authorization data
m      STRING8            authorization protocol name
n      STRING8            authorization protocol data
p                         unused, p=pad(m+n)
4      BITMASK            value-mask (has s bits set to 1)
    #x00000001            timeout
    #x00000002            trust-level
```

```
     #x00000004        group
     #x00000008        event-mask
4s      LISTofVALUE      value-list
```

VALUES

```
4     CARD32       timeout
4                  trust-level
     0             SecurityClientTrusted
     1             SecurityClientUntrusted
4     XID          group
0                  None
4     CARD32          event-mask
     #x00000001      SecurityAuthorizationRevoked
=>
1       1          Reply
1                 unused
2       CARD16     sequence number
4       (q+3)/4      reply length
4       AUTHID      authorization-id
2       CARD16      data-length
18                 unused
q       STRING8     authorization-data-return
r                 unused, r=pad(q)
```

Event Encoding

`SecurityAuthorizationRevoked`

```
1       0+extension event base       code
1                                   unused
2       CARD16                        sequence number
4       AUTHID                        authorization id
24                                   unused
```

Authorization Method Encoding

For authorization-protocol-name "XC-QUERY-SECURITY-1", the authorization-protocol-data is interpreted as follows:

`authorization-protocol-data`

```
1       BITMASK                        policy-mask
       #x00000001          Extensions
       #x00000002          SitePolicies
m       LISTofSECURITYPOLICY                policies
```

`SECURITYPOLICY`

```
1                                           policy-type
```

	0	Permit	
	1	Disallow	
1	CARD8		number of STRs in names
n	LISTofSTR		names

LISTofSTR has the same encoding as in the X protocol: each STR is a single byte length, followed by that many characters, and there is no padding or termination between STRs.

Chapter 6. C Language Binding

The header for this extension is <X11/extensions/security.h>. All identifier names provided by this header begin with XSecurity.

All functions that have return type Status will return nonzero for success and zero for failure.

`Status` **`XSecurityQueryExtension`**`(` *`*dpy`*`,` *`*major_version_return`*`,` *`*minor_version_return`*`);`

`XSecurityQueryExtension` sets major_version_return and minor_version_return to the major and minor Security protocol version supported by the server. If the Security library is compatible with the version returned by the server, it returns nonzero. If dpy does not support the Security extension, or if there was an error during communication with the server, or if the server and library protocol versions are incompatible, it returns zero. No other XSecurity functions may be called before this function. If a client violates this rule, the effects of all subsequent XSecurity calls that it makes are undefined.

`Xauth *`**`XSecurityAllocXauth`**`(`*`void`*`);`

In order to provide for future evolution, Xauth structures are used to pass and return authorization data, and the library provides ways to allocate and deallocate them.

`XSecurityAllocXauth` must be used to allocate the Xauth structure that is passed to `XSecurityGenerateAuthorization`.

For the purposes of the Security extension, the Xauth structure has the following fields:

Type	Field name	Description
unsigned short	name_length	number of bytes in name
char *	name	authorization protocol name
unsigned short	data_length	number of bytes in data
char *	data	authorization protocol data

The Xauth structure returned by this function is initialized as follows: name_length and data_length are zero, and name and data are NULL.

`void` **`XSecurityFreeXauth`**`(` *`*auth`*`);`

`XSecurityFreeXauth` must be used to free Xauth structures allocated by `XSecurityAllocXauth` or returned by `XSecurityGenerateAuthorization`. It is the caller's responsibility to fill in the name and data fields of Xauth structures allocated with `XSecurityAllocXauth`, so this function will not attempt to free them. In contrast, all storage associated with Xauth structures returned from `XSecurityGenerateAuthorization` will be freed by this function, including the name and data fields.

`Bool` **`XSecurityRevokeAuthorization`**`(` *`*dpy`*`,` *`auth_id`*`);`

`XSecurityRevokeAuthorization` deletes the authorization specified by auth_id, which must be a value returned in the auth_id_return parameter of `XSecurityGenerateAuthorization`. All clients that connected with that authorization are be killed. Subsequently, clients that attempt to connect using that authorization will be refused.

`Xauth *`**`XSecurityGenerateAuthorization`**`(` *`*dpy, *auth_in, valuemask, *attributes, *auth_id_return`*`);`

`XSecurityGenerateAuthorization` creates a new authorization with the specified attributes. The auth_in argument must be allocated by `XSecurityAllocXauth`. The name and name_length fields of auth_in should be initialized to the authorization protocol name and its length in characters respectively. If there is authorization data, the data and data_length fields of auth_in should be initialized to the data and its length in characters respectivley. The library does not assume that name and data are null-terminated strings. The auth_in argument must be freed with `XSecurityFreeXauth`.

The XSecurityAuthorizationAttributes structure has the following fields:

Type	**Field name**	**Mask**
unsigned int	trust_level	XSecurityTrustLevel
unsigned int	timeout	XSecurityTimeout
XID	group	XSecurityGroup
long	event_mask	XSecurityEventMask

These correspond to the trust-level, timeout, group, and event-mask described in the SecurityGenerateAuthorization protocol request. The caller can fill in values for any subset of these attributes. The valuemask argument must be the bitwise OR of the symbols listed in the Mask column for all supplied attributes. The event_mask attribute can be None, XSecurityAuthorizationRevokedMask, or XSecurityAllEventMasks. In this revision of the protocol specification XSecurityAllEventMasks is equivalent to XSecurityAuthorizationRevokedMask. If the caller does not need to specify any attributes, the attributes argument can be NULL, and the valuemask argument must be zero.

If the function fails, NULL is returned and auth_id_return is filled in with zero. Otherwise, a pointer to an Xauth structure is returned. The name and name_length fields of the returned Xauth structure will be copies of the name that was passed in, and the data and data_length fields will be set to the authorization data returned by the server. The caller should not assume that name and data are null-terminated strings. If no authorization data was returned by the server, the data and data_length fields will be set to NULL and zero repectively. The returned Xauth structure must be freed with `XSecurityFreeXauth`; the caller should not use any other means free the structure or any of its components. The auth_id_return argument will be filled in with the non-zero authorization id of the created authorization.

The XSecurityAuthorizationRevokedEvent structure has the following fields:

Type	**Field name**	**Description**
int	type	event base + XSecurityAuthorizationRevoked
unsigned long	serial	# of last request processed by server
Bool	send_event	true if this came from SendEvent
Display*	display	Display the event was read from
XSecurityAuthorization	auth_id	revoked authorization id