From	Eric Snowberg <>
Subject	[PATCH v4] certs: Add EFI_CERT_X509_GUID support for dbx entries
Date	Tue, 15 Sep 2020 20:49:27 -0400

```
The Secure Boot Forbidden Signature Database, dbx, contains a list of now
revoked signatures and keys previously approved to boot with UEFI Secure
Boot enabled.  The dbx is capable of containing any number of
EFI_CERT_X509_SHA256_GUID, EFI_CERT_SHA256_GUID, and EFI_CERT_X509_GUID
entries.

Currently when EFI_CERT_X509_GUID are contained in the dbx, the entries are
skipped.

Add support for EFI_CERT_X509_GUID dbx entries. When a EFI_CERT_X509_GUID
is found, it is added as an asymmetrical key to the .blacklist keyring.
Anytime the .platform keyring is used, the keys in the .blacklist keyring
are referenced, if a matching key is found, the key will be rejected.

Signed-off-by: Eric Snowberg <eric.snowberg@oracle.com>
---
v4:
Remove unneeded symbol export found by Jarkko Sakkinen

v3:
Fixed an issue when CONFIG_PKCS7_MESSAGE_PARSER is not builtin and defined
as a module instead, pointed out by Randy Dunlap

v2:
Fixed build issue reported by kernel test robot <lkp@intel.com>
Commit message update (suggested by Jarkko Sakkinen)
---
 certs/blacklist.c                             | 32 +++++++++++++++++++
 certs/blacklist.h                             | 12 +++++++
 certs/system_keyring.c                        |  6 ++++
 include/keys/system_keyring.h                 | 11 +++++++
 .../platform_certs/keyring_handler.c          | 11 +++++++
 5 files changed, 72 insertions(+)

diff --git a/certs/blacklist.c b/certs/blacklist.c
index 6514f9ebc943..4adac7f8fd94 100644
--- a/certs/blacklist.c
+++ b/certs/blacklist.c
@@ -100,6 +100,38 @@ int mark_hash_blacklisted(const char *hash)
 	return 0;
 }
 
+int mark_key_revocationlisted(const char *data, size_t size)
+{
+	key_ref_t key;
+
+	key = key_create_or_update(make_key_ref(blacklist_keyring, true),
+				   "asymmetric",
+				   NULL,
+				   data,
+				   size,
+				   ((KEY_POS_ALL & ~KEY_POS_SETATTR) | KEY_USR_VIEW),
+				   KEY_ALLOC_NOT_IN_QUOTA | KEY_ALLOC_BUILT_IN);
+
+	if (IS_ERR(key)) {
+		pr_err("Problem with revocation key (%ld)\n", PTR_ERR(key));
+		return PTR_ERR(key);
+	}
+
+	return 0;
+}
+
+int is_key_revocationlisted(struct pkcs7_message *pkcs7)
+{
+	int ret;
+
+	ret = validate_trust(pkcs7, blacklist_keyring);
+
+	if (ret == 0)
+		return -EKEYREJECTED;
+
+	return -ENOKEY;
+}
+
 /**
  * is_hash_blacklisted - Determine if a hash is blacklisted
  * @hash: The hash to be checked as a binary blob
diff --git a/certs/blacklist.h b/certs/blacklist.h
index 1efd6fa0dc60..420bb7c86e07 100644
--- a/certs/blacklist.h
+++ b/certs/blacklist.h
@@ -1,3 +1,15 @@
 #include <linux/kernel.h>
+#include <linux/errno.h>
+#include <crypto/pkcs7.h>
 
 extern const char __initconst *const blacklist_hashes[];
+
+#ifdef CONFIG_INTEGRITY_PLATFORM_KEYRING
+#define validate_trust pkcs7_validate_trust
+#else
+static inline int validate_trust(struct pkcs7_message *pkcs7,
+				 struct key *trust_keyring)
+{
+	return -ENOKEY;
+}
+#endif
diff --git a/certs/system_keyring.c b/certs/system_keyring.c
index 798291177186..f8ea96219155 100644
--- a/certs/system_keyring.c
+++ b/certs/system_keyring.c
@@ -241,6 +241,12 @@ int verify_pkcs7_message_sig(const void *data, size_t len,
 			pr_devel("PKCS#7 platform keyring is not available\n");
 			goto error;
 		}
+
+		ret = is_key_revocationlisted(pkcs7);
+		if (ret != -ENOKEY) {
+			pr_devel("PKCS#7 platform key revocationlisted\n");
+			goto error;
+		}
 	}
 	ret = pkcs7_validate_trust(pkcs7, trusted_keys);
 	if (ret < 0) {
diff --git a/include/keys/system_keyring.h b/include/keys/system_keyring.h
index fb8b07daa9d1..b6991cfe1b6d 100644
--- a/include/keys/system_keyring.h
+++ b/include/keys/system_keyring.h
@@ -31,11 +31,14 @@ extern int restrict_link_by_builtin_and_secondary_trusted(
 #define restrict_link_by_builtin_and_secondary_trusted restrict_link_by_builtin_trusted
 #endif
 
+extern struct pkcs7_message *pkcs7;
 #ifdef CONFIG_SYSTEM_BLACKLIST_KEYRING
 extern int mark_hash_blacklisted(const char *hash);
+extern int mark_key_revocationlisted(const char *data, size_t size);
```

```
 extern int is_hash_blacklisted(const u8 *hash, size_t hash_len,
                                const char *type);
 extern int is_binary_blacklisted(const u8 *hash, size_t hash_len);
+extern int is_key_revocationlisted(struct pkcs7_message *pkcs7);
 #else
 static inline int is_hash_blacklisted(const u8 *hash, size_t hash_len,
                                       const char *type)
@@ -47,6 +50,14 @@ static inline int is_binary_blacklisted(const u8 *hash, size_t hash_len)
 {
        return 0;
 }
+static inline int mark_key_revocationlisted(const char *data, size_t size)
+{
+       return 0;
+}
+static inline int is_key_revocationlisted(struct pkcs7_message *pkcs7)
+{
+       return -ENOKEY;
+}
 #endif

 #ifdef CONFIG_IMA_BLACKLIST_KEYRING
diff --git a/security/integrity/platform_certs/keyring_handler.c b/security/integrity/platform_certs/keyring_handler.c
index c5ba695c10e3..cc5a43804bc4 100644
--- a/security/integrity/platform_certs/keyring_handler.c
+++ b/security/integrity/platform_certs/keyring_handler.c
@@ -55,6 +55,15 @@ static __init void uefi_blacklist_binary(const char *source,
        uefi_blacklist_hash(source, data, len, "bin:", 4);
 }

+/*
+ * Revocationlist the X509 cert
+ */
+static __init void uefi_revocationlist_x509(const char *source,
+                                            const void *data, size_t len)
+{
+       mark_key_revocationlisted(data, len);
+}
+
 /*
  * Return the appropriate handler for particular signature list types found in
  * the UEFI db and MokListRT tables.
@@ -76,5 +85,7 @@ __init efi_element_handler_t get_handler_for_dbx(const efi_guid_t *sig_type)
                return uefi_blacklist_x509_tbs;
        if (efi_guidcmp(*sig_type, efi_cert_sha256_guid) == 0)
                return uefi_blacklist_binary;
+       if (efi_guidcmp(*sig_type, efi_cert_x509_guid) == 0)
+               return uefi_revocationlist_x509;
        return 0;
 }
--
2.18.1
```