BPF_LSM + FSVerity for Binary Authorization

Song Liu <song@kernel.org>
Boris Burkov <boris@bur.io>

Binary Authorization

- Only trusted binaries can perform certain operations

- Verify the hash: path based verification is not enough
- Flexible
  - Various access patterns: for example, only signed bpftrace binary can run signed .bt scripts
  - Update allowlist and/or blocklist at runtime
- Small attack surface
  - Use asymmetric keys, so no need to protect the allowlist

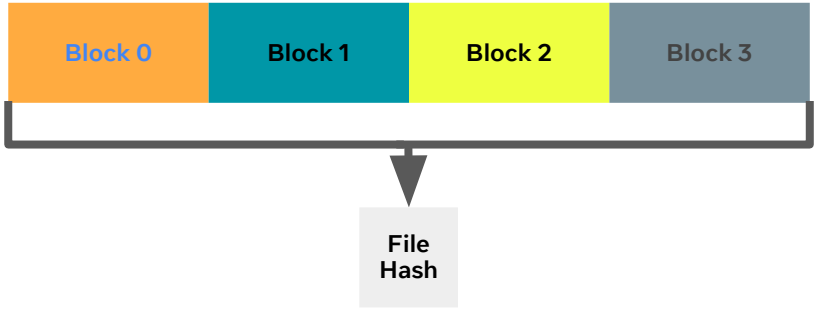
Existing Solutions

- IMA (Integrity Measurement Architecture)
  - Designed for different use cases
  - Not flexible enough
- FSVerity built-in signatures
  - Does not protect metadata
  - Not flexible enough

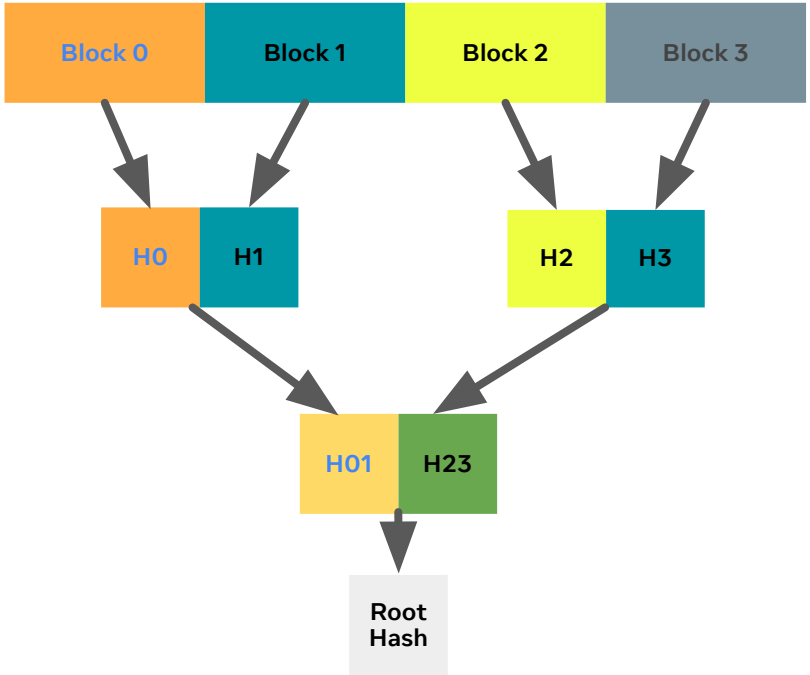
FSVerity

- Provides integrity protection, i.e. detection of accidental (non-malicious) corruption.
- Makes retrieving the file hash extremely efficient.
- Primarily to be used as a tool to support authentication or auditing.
- Enabling verity on a file forces it read-only.
  - open(O_RDWR) will fail regardless of the file mode bits

File Checksum



Merkle tree



Proposed Framework

- FSVerity for file integrity checksums
- Secure binary signing service to compute and sign FSVerity digests
- xattrs to store FSVerity root hash signatures
- BPF_LSM to enforce access control
- User space daemon to manage keyrings and BPF_LSM programs

Example BPF Programs

```
SEC("lsm.s/bprm_creds_from_file")
int BPF_PROG(tag_binary,
           struct linux_binprm *bprm,
           struct file *f)
{
      task = bpf_get_current_task_btf();
      bpf_get_fsverity_digest(f, &digest_ptr);
      bpf_get_file_xattr(f, "user.sig", &sig_ptr);
      ret = bpf_verify_pkcs7_signature(&digest_ptr,
            &sig_ptr, trusted_keyring);
      if (ret)
            return 0;
      bpf_task_storage_get(&allow_list, task,
            NULL, BPF_LOCAL_STORAGE_GET_F_CREATE);
      return 0;
}
```

```
SEC("lsm.s/file_open")
int BPF_PROG(check_file_open, struct
file *f)
{
      if (!is_critical_file(f))
            return 0;
      task =
bpf_get_current_task_btf();
      bpf_task_storage_get(&allowlist,
            task, &value, 0);
      return (value == NULL);
}
```

kfunc `bpf_get_fsverity_digest()`

```
int bpf_get_fsverity_digest(struct file *file,
                            struct bpf_dynptr_kern *digest_ptr)
```

- Only available from LSM hooks
  - No recursion, no deadlock
- `KF_TRUSTED_ARGS`: No pointer walking

kfunc `bpf_get_file_xattr()`

```
int bpf_get_file_xattr(struct file *file, const char *name__str,
                       struct bpf_dynptr_kern *value_ptr);
```

- Only available from LSM hooks
  - No recursion, no deadlock
- `KF_TRUSTED_ARGS`: No pointer walking
- Can only access `user.*` xattrs
  - `security.bpf` namespace might be required

Summary

- Latest patchset: https://lore.kernel.org/bpf/20231104001313.3538201-1-song@kernel.org/

Thank You!