One Identity Safeguard for Privileged Sessions Security Briefing

Security Hardening

Our Solution is based on Zorp OS, which is a derivate of Ubuntu Linux LTS, we did the hardening of it based on the official Ubuntu guides. Please find the following extra security features in our solution enabled:

Stack Protector

gcc's -fstack-protector provides a randomized stack canary that protects against stack overflows, and reduces the chances of arbitrary code execution via controlling return address destinations. Enabled at compile-time. (A small number of applications do not play well with it, and have it disabled.) The routines used for stack checking are actually part of glibc, but gcc is patched to enable linking against those routines by default.

Heap Protector

The GNU C Library heap protector (both automatic via ptmalloc and manual) provides corrupted-list/unlink/double-free/overflow protections to the glibc heap memory manager (first introduced in glibc 2.3.4). This stops the ability to perform arbitrary code execution via heap memory overflows that try to corrupt the control structures of the malloc heap memory areas.
This protection has evolved over time, adding more and more protections as additional corner-cases were researched. As it currently stands, glibc 2.10 and later appears to successfully resist even these hard-to-hit conditions.

Pointer Obfuscation

Some pointers stored in glibc are obfuscated via PTR_MANGLE/PTR_UNMANGLE macros internally in glibc, preventing libc function pointers from being overwritten during runtime.

Address Space Layout Randomisation (ASLR)

ASLR is implemented by the kernel and the ELF loader by randomising the location of memory allocations (stack, heap, shared libraries, etc). This makes memory addresses harder to predict when an attacker is attempting a memory-corruption exploit. ASLR is controlled system-wide by the value of /proc/sys/kernel/randomize_va_space. This includes brk ASLR, it defaults to "2" (on, with brk ASLR).
(STACK ASLR, LIBS/MMAP ASLR, EXEC ASLR, BRK ASLR, VDSO ASLR) This means that practically everything is randomized.

Built as PIE

All programs built as Position Independent Executables (PIE) with "-fPIE -pie" can take advantage of the exec ASLR. This protects against "return-to-text" and generally frustrates memory corruption attacks. This requires centralized changes to the compiler options when building the entire archive. PIE has a large (5-10%) performance penalty on architectures with small numbers of general registers (e.g. x86), so it should only be used for a select number of security-critical packages (some upstreams natively support building with PIE, other require the use of "hardening-wrapper" to force on the correct compiler and linker flags).

Non-Executable Memory

Most modern CPUs protect against executing non-executable memory regions (heap, stack, etc). This is known either as Non-eXecute (NX) or eXecute-Disable (XD). This protection reduces the areas an attacker can use to perform arbitrary code execution. It requires that the kernel use "PAE" addressing (which also allows addressing of physical addresses above 3GB). The 64bit and 32bit -server and -generic-pae kernels are compiled with PAE addressing. Starting in Ubuntu 9.10, this protection is partially emulated for processors lacking NX when running on a 32bit kernel (built with or without PAE). After booting, you can see what NX protection is in effect:

- Hardware-based (via PAE mode):

```
[ 0.000000] NX (Execute Disable) protection: active
```

- Partial Emulation (via segment limits):

```
[ 0.000000] Using x86 segment limits to approximate NX protection
```

/dev/mem protection

Some applications (Xorg) need direct access to the physical memory from user-space. The special file /dev/mem exists to provide this access. In the past, it was possible to view and change kernel memory from this file if an attacker had root access. The CONFIG_STRICT_DEVMEM kernel option was introduced to block non-device memory access (originally named CONFIG_NONPROMISC_DEVMEM).

/dev/kmem disabled

There is no modern user of /dev/kmem any more beyond attackers using it to load kernel rootkits. CONFIG_DEVKMEM is set to "n".

/proc/$pid/maps protection

With ASLR, a process's memory space layout suddenly becomes valuable to attackers. The "maps" file is made read-only except to the process itself or the owner of the process. Went into mainline kernel with sysctl toggle in 2.6.22. The toggle was made non-optional, forcing the privacy to be enabled regardless of sysctl settings.

Symlink restrictions

A long-standing class of security issues is the symlink-based ToCToU race, most commonly seen in world-writable directories like /tmp/. The common method of exploitation of this flaw is crossing privilege boundaries when following a given symlink (i.e. a root user follows a symlink belonging to another user).
In Ubuntu 10.10 and later, symlinks in world-writable sticky directories (e.g. /tmp) cannot be followed if the follower and directory owner do not match the symlink owner.

Hardlink restrictions

Hardlinks can be abused in a similar fashion to symlinks above, but they are not limited to world-writable directories. If /etc/ and /home/ are on the same partition, a regular user can create a hardlink to /etc/shadow in their home directory. While it retains the original owner and permissions, it is possible for privileged
programs that are otherwise symlink-safe to mistakenly access the file through its hardlink. Additionally, a very minor untraceable quota-bypassing local denial of service is possible by an attacker exhausting disk space by filling a world-writable directory with hardlinks.
Hardlinks cannot be created to files that the user would be unable to read and write originally, or are otherwise sensitive.

Restricted ptrace

A troubling weakness of the Linux process interfaces is that a single user is able to examine the memory and running state of any of their processes. For example, if one application was compromised, it would be possible for an attacker to attach to other running processes (e.g. SSH sessions, GPG agent, etc) to extract additional credentials and continue to immediately expand the scope of their attack without resorting to user-assisted phishing or trojans.
In Ubuntu 10.10 and later, users cannot ptrace processes that are not a descendant of the debugger.

SYN cookies enabled

SYN cookie use is activated, which helps mitigate a SYN-flood attack.

Access to Local Console

The console access with the root account is restricted for deep troubleshooting or extra-ordinary maintenance purposes only. No any console actions should be made without direct instructions of the One Identity Support Team. The appliance can be sealed to avoid remote SSH access to the console:
https://support.oneidentity.com/technical-documents/one-identity-safeguard-for-privileged-sessions/5.10.0/administration-guide/38#TOPIC-1090705

Local Users with Configured Shell

The only functional ID on the system is root. The postgres user has no password on the system, hence it is harmless. It has been created due to the internal behaviour of PostgreSQL, which is used internally the appliance.

Encryption Algorithms, Key Management

Box management/Servers

Nginx

- TLS cipher suite: HIGH:!aNULL:!MD5;
- TLSv1.2

SSH

- Ciphers aes256-ctr,aes128-ctr
- MACs hmac-sha2-512,hmac-sha2-256
- KexAlgorithms diffie-hellman-group-exchange-sha256

SNMP

With SNMPv3 agent, authentication uses SHA1 or MD5. For encryption of communication it uses AES-128-CFB or DES (What NetSNMP offers)

Box management/Clients

The algorithms used here depends mainly on the server side configuration, not the client side.

Proxies

We use what the protocol offers, in case of SSH, client and server cipher suites are configurable: https://support.oneidentity.com/technical-documents/one-identity-safeguard-for-privileged-sessions/5.10.0/administration-guide/57#TOPIC-1090782
Telnet and VNC TLS cipher suites are configurable.
Starting with version 5.11, the minimum TLS version is configurable in telnet, VNC, HTTP and RDP protocols. The selection of the available cipher suites are simplified to "recommended" and "custom" options.

Audit trails and sensitive assets

Audit trail encryption

- SHA2-512 for key and IV generation from random bytes
- RSA encryption with a user supplied key, used with PKCS#1 v1.5 padding
- AES-128-GCM for symmetric encryption
- padding with 0s

Audit trail digital signature

- SHA2-512 for message digest
- RSA digital signature with user supplied key, used with PKCS#1 v1.5 padding **OR** DSA digital signature with user supplied key

Screenshot encryption

- RSA encryption with a user supplied key, used with OAEP padding
- AES-128-CBC for symmetric encryption
- random bytes from a PRNG are used for IV
- padding: no padding is used (CTS aka ciphertext stealing is used, therefore the length of the plaintext and ciphertext are equal)

Video encryption

- RSA encryption with a user supplied key, used with OAEP padding
- AES-128-ECB for symmetric encryption
- padding: PKCS#7

Private Keystore (in case of storing keys permanently)

- SHA1 for key and IV generation from random bytes
- AES-256-CBC for symmetric encryption
- padding: PKCS#5 (see openssl enc documentation)

Credential stores (in case of encryption with passwords):

- AES-256-CBC for symmetric encryption
- padding: PKCS#7
- IV is 0

Config export (in case of encryption with passwords):

- AES-256-CBC for symmetric encryption
- SHA1 for key derivation and IV generation from the password
- padding: PKCS#5 (see openssl enc documentation)

Config export (in case of GPG encryption):

- Depends on the preferred algorithms listed in the user supplied GPG key.

Key management

There are no keys 'managed' on the system, just stored in the configuration for certain functionality. For certain cases, certificates may be generated on the box either on-the-fly or for static usage. In such case the necessary private keys are also generated. (see below). For the same purpose, externally generated certificates or keys are accepted to upload to the system. Depending on the configuration (i.e. key based authentication in SSH), externally generated public and private keys may be uploaded to the system.

In case of externally generated certificates and keys, they must comply with the following format constraints: https://support.oneidentity.com/technical-documents/one-identity-safeguard-for-privileged-sessions/5.10.0/administration-guide/39#TOPIC-109071 Private keys are required along with the respective certificates in the solution for TLS support.

The certificates and the corresponding private keys can be generated statically, or on-the-fly (for supporting TLS in an encrypted audited connection). On-the-fly key generation is made by the proxy engine via OpenSSL. For such case, the system can generate self-signed CA certificates upon the details given in the configuration as described at: https://support.oneidentity.com/technical-documents/one-identity-safeguard-for-privileged-sessions/5.10.0/administration-guide/39#TOPIC-1090711

CA certificates can also be generated externally via a plugin for the purpose of generating certificate and private key pairs signed by a Certificate Authority. By using this type of plugin the certificate signing can be tailored to fit any custom requirement: https://support.oneidentity.com/technical-documents/one-identity-safeguard-for-privileged-sessions/5.10.0/administration-guide/45#TOPIC-1090733 Both static and on-the fly certificates on the system are signed by the designated CA certificate stored in the configuration (as created by any of the above methods).

Key generation also happens when clustering is used for the purpose of central management and search. Nodes in the cluster connect to each other using IPsec, where PSK is generated via the random.SystemRandom().choice method of python3, working with /dev/urandom.

Key distribution

Key distribution only happens at building a cluster of nodes for the purpose of central management and central search. The generated key is encoded into the token, as described at https://support.oneidentity.com/technical-documents/one-identity-safeguard-for-privileged-sessions/5.10.0/administration-guide/31#TOPIC-1090679. Other than that, keys or certificates are not distributed, but stored in the configuration for the usage with the configured functionality. The public parts of the keys can be downloaded from the system via its web GUI with the appropriate rights in PEM, DER, OpenSSH format.

When the configuration is exported, it can be encrypted as:

- at manual export with GPG key or with password
- at system backup it can be encrypted with GPG key

Keys stored on appliance

Inside the system keys are not encrypted, except when storing in the Credential Store (used for SSH auto-login for the monitored connections) or when stored in the private keystore of the user (used for the purpose of being able to replay encrypted session recordings).

The system also has an external component that indexes the textual content of the recorded sessions: https://support.oneidentity.com/technical-documents/one-identity-safeguard-for-privileged-sessions/5.10.0/administration-guide/60#TOPIC-1090796. The external indexer is a Linux service component that processes the recorded audit trails and sends indexed data back to the appliance via secure channel. For being able to process encrypted audit trails, the indexer services must have access to the corresponding private keys. To secure these keys, it is possible to configure a hardware security module (HSM) or smart card to integrate with external indexer as described at https://support.oneidentity.com/technical-documents/one-identity-safeguard-for-privileged-sessions/5.10.0/administration-guide/62#TOPIC-1090804

Event Logging

The system logs are written into local Syslog, accessible through the web GUI at Basic Settings > Troubleshooting > View Log Files. Available methods are Download, View, or Tail. There is a separate log file created each day and logs are deleted after 1 week from the appliance.

Configuration changes are also written into a separate database, accessible on the web GUI at AAA > Accounting.

Session related audit information are written into the meta-database and accessible through the Search interface.

There are 2 ways of forwarding log events to an external system:

- The generic syslog forward functionality: https://support.oneidentity.com/technical-documents/one-identity-safeguard-for-privileged-sessions/5.10.0/administration-guide/16#TOPIC-1090610
- The universal SIEM forwarder, via syslog in special JSON (for Splunk) or CEF format: https://support.oneidentity.com/technical-documents/one-identity-safeguard-for-privileged-sessions/5.10.0/administration-guide/22#TOPIC-1090644

The universal SIEM forwarder is for the purpose of ingesting session related information to SIEM solutions. There are 3 distinct types of events generated and forwarded through the SIEM forwarder: content messages, meta messages, score messages. The first 2 is related to SPS/PSM, the last is related to the SPA, behavioral analytics module. The certain Events of each type are described at https://support.oneidentity.com/technical-documents/one-identity-safeguard-for-privileged-sessions/5.10.0/administration-guide/22#TOPIC-1090644

The generic Syslog output contains lot more information (basically all the system logs), but also contains logs overlapping with the events on the SIEM forwarder. In general, the overlapping parts are the meta messages. The generic Syslog contains lot more information about the session too, some containing security related information.