

LPIC-3
(303-300)



LPIC-3 REFERENCE

Study Guide ^{1st} Edition

معرفی کوتاه کتاب

گواهینامه LPIC-3 بالاترین سطح از مجموعه گواهینامه‌های حرفه‌ای چندسطحی موسسه لینوکس (LPI) است. این گواهینامه برای متخصصان لینوکس در سطح سازمانی طراحی شده و معتبرترین مدرک حرفه‌ای لینوکس، به صورت مستقل از توزیع خاص، در صنعت به شمار می‌رود. در حال حاضر چهار شاخه تخصصی از LPIC-3 ارائه می‌شود که هرکدام حوزه‌ای مشخص را پوشش می‌دهند. در این کتاب تمرکز بر روی شاخه LPIC-3 Security است؛ شاخه‌ای که به مدیریت سیستم‌های لینوکس در مقیاس سازمانی با محوریت امنیت می‌پردازد. مطالعه این کتاب به شما کمک می‌کند تا برای آزمون 303 (نسخه 3.0) آمادگی کامل کسب کنید.

کتاب

مرجع

LPIC1

لینوکس

کتاب مرجع کامل مدرک بین المللی

LPIC-3 Security

(303)

Linux Professional Institute Certification Study Guide

حسین سیلانی

ویرایش اول

۱۴۰۴



سرشناسه	سیلانی، حسین، ۱۳۶۰.
عنوان و نام پدیدآور	کتاب مرجع کامل مدرک بین المللی Ipic 3 303
مشخصات نشر	حسین سیلانی؛ ویراستاری حسین سیلانی.
مشخصات ظاهری	تهران: کیان دانش، ۱۴۰۴.
شابک	600 ص.
وضعیت فهرست نویسی	۶۰۵۸۱، ۶۲۲، ۴۰۱، ۹۷۸، ۶۰۵۸۱
موضوع	فیپا لینوکس Linux
شناسه افزوده	سیلانی، حسین، ۱۳۶۰.
رده بندی کنگره	۷۶/۷۶QA
رده بندی دیویی	۴۳۲/۰۰۵
شماره کتابشناسی ملی	۹۶۷۳۵۹۳
اطلاعات رکورد کتابشناسی	فیپا

نشر کیان دانش: ناشر کتاب‌های دانشگاهی

عنوان: کتاب مرجع کامل مدرک بین المللی Ipic 3 303

تألیف: حسین سیلانی

صفحه آرایی و ویراستاری: حسین سیلانی

چاپ و صحافی: موسسه مهراد

طراحی جلد: حسین سیلانی

چاپ اول: ۱۴۰۴

شمارگان: ۱۰۰ جلد

شابک: ۶۰۵۸۱، ۴۰۱، ۶۲۲، ۹۷۸، ۶۰۵۸۱

حق چاپ محفوظ است.



خیابان ۱۲ فروردین، خیابان شهدای ژاندارمری، شماره ۱۲۶، طبقه چهارم

سخنی از نویسنده
با سلام و احترام به دوست گرامی

از اینکه این کتاب را انتخاب کرده‌اید و در این مسیر همراه من هستید صمیمانه سپاسگزارم. امیدوارم این اثر برای علاقه‌مندان به حوزه لینوکس مفید و اثربخش باشد. همچنین از شما درخواست دارم که اگر نواقص یا کاستی‌هایی در کتاب مشاهده کردید آن‌ها را نادیده نگیرید و با ارائه نظرات و پیشنهادات سازنده‌ی خود، به بهبود کیفیت این اثر کمک کنید.

شایان ذکر است که این کتاب یکی از صدها کتاب لینوکسی من است که بسیاری از آن‌ها در حال ترجمه یا نگارش هستند. این مجموعه در راستای پروژه‌های متن‌باز شخصی‌ام و با هدف طراحی و توسعه توزیع‌های لینوکس نوشته شده است. امیدوارم این تلاش‌ها گامی هرچند کوچک در جهت گسترش دانش و استفاده از سیستم‌عامل‌های متن‌باز باشد.

 <https://predator.os.ir/>

 <https://littlePsycho.ir>

 <https://emperor.os.ir/>

 <https://ubuntu.ir>

تصمیم دارم محتوای آموزشی فارسی در حوزه‌های لینوکس و نرم‌افزارهای متن‌باز را در قالب‌های متنوعی مانند کتابچه، کتاب، فلش‌کارت، فایل‌های صوتی و ویدئوهای آموزشی تولید کرده و از طریق یک سایت در دسترس عموم قرار دهم. هدف من از این کار، ایجاد یکی از بزرگترین و جامع‌ترین مراجع آموزشی فارسی در این زمینه است. برای دسترسی به سایت و اطلاع از آخرین اخبار و محتواهای آموزشی، می‌توانید از طریق راه‌های ارتباطی زیر با من در تماس باشید:

 learninghive.ir: آدرس سایت

 [@LinuxTNT](https://t.me/LinuxTNT): کانال تلگرام

 [@seilany](https://t.me/seilany): آیدی تلگرام

 h.seilany@gmail.com: ایمیل

 [hosseinseilani](https://github.com/hosseinseilani): گیت‌هاب

منتظر حضور و همراهی شما هستم!

تقديم به تو

يگانه فرزانه دلم

فهرست مطالب

مقدمه	- ۲۶ -
فصل ۱	- ۳۱ -
۱، ۳۳۱ گواهی نامه‌های X.509 و زیرساخت کلید عمومی (PKI)	- ۳۱ -
موضوع ۳۳۱: رمزنگاری	- ۳۱ -
مفاهیم پایه رمزنگاری	- ۳۲ -
چرخه زندگی یک گواهی نامه X.509	- ۳۲ -
زنجیره اعتماد (Trust Chain)	- ۳۲ -
ابزار OpenSSL	- ۳۳ -
سرویس‌های خودکار صدور گواهی	- ۳۳ -
کاربردهای رمزنگاری	- ۳۳ -
عناصر اصلی رمزنگاری	- ۳۴ -
کلید (Key)	- ۳۴ -
الگوریتم (Algorithm)	- ۳۵ -
متن ساده و متن رمز شده	- ۳۵ -
رمزگذاری متقارن و نامتقارن	- ۳۶ -
۱. رمزگذاری متقارن Symmetric Encryption	- ۳۶ -
۲. رمزگذاری نامتقارن Asymmetric Encryption	- ۳۷ -
یکپارچگی داده‌ها از طریق هش‌ها (Data Integrity through Hashes)	- ۳۷ -
الگوریتم‌های رایج هش	- ۳۹ -
CRC32 – Cyclic Redundancy Check	- ۳۹ -
MD5 – Message Digest 5	- ۳۹ -
SHA-1 – Secure Hash Algorithm 1	- ۴۰ -
SHA-2 و SHA-3	- ۴۰ -
نحوه عملکرد هش	- ۴۰ -
نقش Salt در هش	- ۴۱ -
اهمیت هش در امنیت اطلاعات	- ۴۲ -
زیرساخت کلید عمومی (PKI) و زنجیره‌های اعتماد	- ۴۲ -
PKI چیست؟	- ۴۲ -
مرجع صدور گواهی (Certificate Authority – CA) چیست؟	- ۴۳ -
چرا CA اهمیت دارد؟	- ۴۳ -
فرآیند صدور گواهی توسط CA	- ۴۳ -
تشبیه CA با اداره پاسپورت	- ۴۴ -
اهمیت گواهی‌های دیجیتال	- ۴۴ -
زنجیره اعتماد (Trust Chain)	- ۴۴ -
چرا زنجیره اعتماد مهم است؟	- ۴۵ -
تشبیه زنجیره اعتماد به ساختار خانوادگی	- ۴۵ -

- ۴۵ - پیامدهای شکستن زنجیره اعتماد
- ۴۵ - فرآیند صدور گواهی
- ۴۶ - درخواست صدور گواهی دیجیتال Certificate Signing Request -
- ۴۷ - نحوه عملکرد مرجع صدور گواهی Certificate Authority -
- ۴۷ - اهمیت CSR و CA در امنیت اینترنت و مدیریت گواهی ها
- ۴۷ - فرآیند ایجاد CSR و صدور گواهی توسط CA
- ۴۸ - اهمیت اعتماد به CA
- ۴۸ - چالش های امنیتی در مدیریت گواهی ها
- ۴۸ - فهرست لغو گواهی ها CRL
- ۴۸ - پروتکل وضعیت آنلاین گواهی OCSP -
- ۴۹ - مقایسه CRL و OCSP
- ۴۹ - ایجاد و کار با گواهی های دیجیتال در لینوکس
- ۴۹ - ایجاد کلید خصوصی Private Key -
- ۵۰ - نمایش اطلاعات گواهی Display Certificate -
- ۵۱ - ایجاد درخواست صدور گواهی CSR -
- ۵۲ - فرمت فایل های گواهی X.509
- ۵۲ - فرمت PEM - Privacy Enhanced Mail -
- ۵۲ - فرمت DER - Distinguished Encoding Rules -
- ۵۲ - فرمت PKCS#7
- ۵۳ - فرمت PKCS#12
- ۵۴ - تبدیل فرمت ها با OpenSSL
- ۵۵ - تبدیل بین فرمت های گواهی با OpenSSL
- ۵۵ - مدیریت یک مرجع صدور گواهی
- ۵۵ - پیکربندی OpenSSL و ایجاد کلید خصوصی
- ۵۶ - امضای CSR توسط CA
- ۵۶ - فایل های ضروری برای CA OpenSSL
- ۵۷ - ابزار جایگزین برای مدیریت کلیدها

فصل ۲ - ۵۹ - گواهی های X.509 برای رمزنگاری، امضا و احراز هویت - ۵۹ - X.509 Certificates for Encryption, Signing and Authentication - ۵۹ -

- ۶۰ - پیاده سازی HTTPS با Apache HTTPD
- ۶۰ - استفاده از OpenSSL برای تست SSL/TLS
- ۶۱ - اهمیت عملی X.509 برای سرورها و کلاینت ها
- ۶۱ - SSL، TLS و تهدیدات رایج امنیتی لایه انتقال
- ۶۲ - تفاوت های SSL و TLS
- ۶۲ - کاربرد SSL و TLS در شبکه
- ۶۲ - تهدیدات رایج امنیتی در لایه انتقال

۶۳	-----	پایاده سازی عملی TLS و HTTPS در Apache HTTPD با استفاده از گواهی های X.509
۶۵	-----	ایجاد یک مرجع صدور گواهی داخلی
۶۸	-----	صدور گواهی کلاینت Client Certificates
۶۹	-----	تنظیم Apache HTTPD برای احراز هویت کلاینت
۷۰	-----	مزایای Apache TLS Mutual Authentication در
۷۰	-----	بررسی زنجیره اعتماد با OpenSSL
۷۱	-----	امنیت لایه انتقال Transport Layer Security -
۷۲	-----	نحوه عملکرد TLS - How TLS Works
۷۴	-----	فرایند TLS/SSL Handshake
۷۵	-----	حمله مرد میانی - تعریف و مفهوم کلی
۷۶	-----	مکانیسم ها و شکل های رایج MITM
۷۶	-----	نمونه های تاریخی و آسیب پذیری های شناخته شده مرتبط
۷۶	-----	علائم و روش های تشخیص حمله مرد میانی
۷۷	-----	پیشگیری: اصول و پایاده سازی های مؤثر در سطح کلاینت
۷۷	-----	پیشگیری: اقدامات شبکه ای و ساختاری در سطح سازمانی
۷۸	-----	پیشگیری: مدیریت گواهی ها و پایاده سازی PKI محکم
۷۸	-----	تشخیص پیشرفته و واکنش به حادثه (Incident Response)
۷۸	-----	ابزارها و آموزش: تمرین ایمن و تحلیل دفاعی
۷۸	-----	کنترل های فنی تکمیلی و بهترین شیوه ها Checklist عملی
۷۹	-----	کار با mod_ssl در وب سرور آپاچی
۷۹	-----	محل و ساختار فایل پیکربندی mod_ssl
۸۰	-----	دستورات کلیدی در ssl.conf
۸۱	-----	ارتباط mod_ssl با OpenSSL
۸۱	-----	تقویت الگوریتم های رمزنگاری Cipher Strength -
۸۲	-----	پیکربندی گواهی های سرور Server Certificates
۸۳	-----	احراز هویت کلاینت با گواهی دیجیتال
۸۳	-----	پیکربندی Apache برای Client Certificate Authentication
۸۴	-----	مزایای استفاده از Client Certificate Authentication
۸۴	-----	چالش های پایاده سازی Client Certificate Authentication
۸۵	-----	غیرفعال کردن پروتکل های SSLv2 و SSLv3 در Apache
۸۵	-----	آسیب پذیری های SSLv2 و SSLv3
۸۵	-----	پیکربندی Apache برای غیرفعال کردن SSLv2 و SSLv3
۸۶	-----	تمرکز روی TLS 1.2 و TLS 1.3
۸۶	-----	بررسی و تست پس از پیکربندی
۸۶	-----	تقویت الگوریتم های رمزنگاری Cipher Strength - در mod_ssl Apache
۸۶	-----	پیکربندی پایه Cipher Suite
۸۷	-----	مثال پیکربندی امن برای TLS 1.2 و TLS 1.3
۸۸	-----	بررسی Cipher Suite های فعال

- ۸۸ - _____ غیرفعال کردن پروتکل های قدیمی SSL و مدیریت امنیت در لینوکس
- ۸۹ - _____ تاریخچه SSL و تکامل به TLS
- ۸۹ - _____ اهمیت غیرفعال کردن پروتکل های قدیمی
- ۸۹ - _____ مثال پیکربندی در Apache
- ۹۰ - _____ مثال پیکربندی در Nginx
- ۹۰ - _____ معرفی حملات شناخته شده علیه پروتکل های قدیمی
- ۹۰ - _____ حمله POODLE
- ۹۰ - _____ حمله BEAST
- ۹۰ - _____ حمله CRIME
- ۹۰ - _____ حملات Downgrade
- ۹۰ - _____ جایگزین های امن TLS 1.2 و TLS 1.3
- ۹۱ - _____ Apache در OSCP Stapling
- ۹۱ - _____ OSCP چیست؟
- ۹۱ - _____ OSCP Stapling
- ۹۲ - _____ پیکربندی OSCP Stapling در Apache
- ۹۲ - _____ چرا OSCP Stapling اهمیت دارد؟
- ۹۳ - _____ نکات عملی هنگام استفاده از OSCP Stapling
- ۹۵ - _____ شناسایی نام سرور Server Name Identification (SNI)
- ۹۶ - _____ مشکل میزبانی چندین سایت روی یک IP
- ۹۶ - _____ راه حل SNI
- ۹۶ - _____ تاریخچه و پشتیبانی Apache
- ۹۶ - _____ دستور SSLStrictSNIVHostCheck در Apache
- ۹۷ - _____ نحوه عملکرد SSLStrictSNIVHostCheck
- ۹۷ - _____ پیکربندی در Apache
- ۹۸ - _____ مزایای SNI
- ۹۸ - _____ چرا به HSTS نیاز داریم؟
- ۹۹ - _____ نحوه عملکرد HSTS
- ۹۹ - _____ اجزای هدر HSTS
- ۱۰۰ - _____ چرا ابتدا باید max-age کوتاه باشد؟
- ۱۰۰ - _____ پیش نیازهای فعال سازی HSTS
- ۱۰۰ - _____ مزایای HSTS
- ۱۰۱ - _____ پیکربندی ریدایرکت به HTTPS
- ۱۰۱ - _____ ترکیب ریدایرکت با HSTS
- ۱۰۱ - _____ خطایابی با OpenSSL

فصل ۳ - ۱۰۴ - _____

۳۳۱.۳ سیستم فایل های رمزگذاری شده (Encrypted File Systems) - ۱۰۴ - _____

- ۱۰۴ - _____ مفاهیم رمزگذاری دیسک (Disk Encryption Concepts)

۱۰۵ -	روش های رمزگذاری دیسک
۱۰۵ -	رمزگذاری کامل دیسک (Whole Disk Encryption)
۱۰۵ -	رمزگذاری در سطح فایل سیستم
۱۰۵ -	ابزارهای رمزگذاری دیسک
۱۰۶ -	رمزگذاری در سطح Block Device
۱۰۶ -	آشنایی با LUKS (Linux Unified Key Setup)
۱۰۷ -	کار با LUKS در عمل
۱۰۸ -	استفاده از کلید برای باز کردن خودکار در زمان بوت
۱۰۹ -	فایل crypttab
۱۰۹ -	رمزگذاری در لینوکس با LUKS ، crypttab ، eCryptfs و EncFS
۱۰۹ -	فایل crypttab و نقش آن در سیستم های LUKS
۱۰۹ -	نحوه ایجاد crypttab
۱۱۰ -	اتصال crypttab به fstab
۱۱۰ -	چرایی اهمیت crypttab
۱۱۰ -	رمزگذاری سطح فایل سیستم
۱۱۰ -	eCryptfs (Enterprise Cryptographic Filesystem)
۱۱۰ -	نصب eCryptfs
۱۱۱ -	نحوه استفاده
۱۱۱ -	مدیریت کلیدها در eCryptfs
۱۱۱ -	ecryptfs-add-passphrase
۱۱۲ -	ecryptfs-manager
۱۱۲ -	ecryptfs-stat
۱۱۲ -	یکپارچگی با PAM
۱۱۲ -	EncFS
۱۱۳ -	مدیریت EncFS
۱۱۳ -	مقایسه eCryptfs و EncFS
۱۱۳ -	مثال های کاربردی از cryptsetup و LUKS

فصل ۴ - ۱۱۷ -

۱۱۷ - 331.4 DNS و رمزنگاری

۱۱۷ -	آشنایی با مفاهیم پایه DNS
۱۱۸ -	مفاهیم اصلی DNSSEC
۱۱۸ -	بیکربندی و عیب یابی BIND برای سرو DNSSEC
۱۱۹ -	بیکربندی BIND به عنوان Recursive Resolver با اعتبارسنجی DNSSEC
۱۱۹ -	CAA و DANE
۱۱۹ -	رکورد CAA
۱۱۹ -	DANE
۱۱۹ -	Transaction Signatures یا TSIG

تکنولوژی‌های جدید در DNS	۱۱۹ -
فایل‌ها، دستورات و ابزارهای مهم در مدیریت DNS و DNSSEC	۱۲۰ -
کار با DNS	۱۲۰ -
انواع سرورهای DNS	۱۲۰ -
سرور مرجع Authoritative DNS Server	۱۲۰ -
سرور غیر مرجع Non-Authoritative DNS Server	۱۲۱ -
رزولورها (Resolvers)	۱۲۱ -
رزولور بازگشتی Recursive Resolver	۱۲۱ -
رزولور تکراری (Iterative Resolver)	۱۲۱ -
زون‌ها و رکوردهای DNS	۱۲۱ -
زون عمومی Public Zone	۱۲۲ -
زون خصوصی Private Zone	۱۲۲ -
رکوردهای DNS در زون‌ها	۱۲۲ -
اهمیت امنیت در مدیریت زون‌ها	۱۲۳ -
زیرزون تفویض شده Delegated Subzone	۱۲۳ -
DNS چندافاق Split-horizon DNS	۱۲۴ -
مزایای Split-horizon DNS	۱۲۴ -
رکوردهای منبع (Resource Records – RR)	۱۲۵ -
ساختار رکوردها	۱۲۵ -
انواع متداول رکوردهای DNS	۱۲۵ -
بخش اول: مجموعه رکوردها (Record Sets)	۱۲۶ -
امنیت در BIND (Securing BIND)	۱۲۷ -
امضاهای تراکنشی (TSIG)	۱۲۷ -
اجرای BIND در محیط Chroot Jail	۱۲۸ -
فایل پیکربندی named.conf	۱۲۸ -
ساختار نمونه فایل named.conf	۱۳۰ -
ابزار مدیریت BIND با rndc	۱۳۱ -
بازسازی فایل rndc.key	۱۳۲ -
تفاوت استفاده از reload و restart	۱۳۲ -
قابلیت‌های پیشرفته rndc	۱۳۳ -
اهمیت امنیتی rndc	۱۳۳ -
ابزار dig در لینوکس	۱۳۳ -
گزینه‌های پر کاربرد در dig	۱۳۳ -
ابزار delv در BIND	۱۳۴ -
مقایسه dig و delv	۱۳۴ -
ساختار دستور delv	۱۳۴ -
گزینه‌های مهم delv	۱۳۵ -

- ۱۳۵ - ----- گزینه‌های اشکال‌زدایی در delv
- ۱۳۵ - ----- DNSSEC
- ۱۳۶ - ----- رکوردهای جدید در DNSSEC
- ۱۳۶ - ----- RRsets
- ۱۳۶ - ----- کلیدهای امضا KSK و ZSK
- ۱۳۶ - ----- فرآیند اعتبارسنجی

فصل ۵ ----- ۱۴۰ -

332.1 سخت‌سازی میزبان (Host Hardening) موضوع ۳۳۲: امنیت میزبان (Host Security) ----- ۱۴۰ -

- ۱۴۰ - ----- امنیت کرنل (Kernel Security)
- ۱۴۱ - ----- غیرفعال کردن نرم‌افزارها و سرویس‌های غیرضروری
- ۱۴۱ - ----- محدود کردن استفاده از منابع سیستم
- ۱۴۳ - ----- سناریوهای واقعی استفاده از محدودیت‌ها
- ۱۴۳ - ----- تنظیم و بهینه‌سازی پارامترهای کرنل لینوکس
- ۱۴۳ - ----- ابزار کلیدی: دستور sysctl
- ۱۴۳ - ----- نمایش پارامترها: کاوش در تنظیمات کرنل
- ۱۴۴ - ----- تغییر پارامترها به صورت بلادرنگ
- ۱۴۴ - ----- ذخیره‌سازی تغییرات به صورت دائمی
- ۱۴۵ - ----- پشت پرده: سیستم فایل (/proc/sys) procfs
- ۱۴۵ - ----- تحلیل جزئی پارامترهای ICMP یک مطالعه موردی
- ۱۴۶ - ----- قابلیت‌های لینوکس و ضرورت وجود آن‌ها
- ۱۴۷ - ----- مفهوم Linux Capabilities
- ۱۴۷ - ----- معماری قابلیت‌ها: مجموعه‌های پنج‌گانه
- ۱۴۷ - ----- مجموعه مجاز (Permitted Capabilities – CapPrm)
- ۱۴۸ - ----- مجموعه مؤثر (Effective Capabilities – CapEff)
- ۱۴۸ - ----- مجموعه به ارث‌رسیدنی (Inheritable Capabilities – CapInh)
- ۱۴۸ - ----- مجموعه مرزی (Bounding Set – CapBnd)
- ۱۴۸ - ----- مجموعه محیطی (Ambient Capabilities – CapAmb)
- ۱۴۹ - ----- مثال عملی: بررسی قابلیت‌های یک فرآیند
- ۱۴۹ - ----- مدیریت قابلیت‌ها با Setcap و Getcap
- ۱۴۹ - ----- تأثیر Capabilities بر امنیت سیستم
- ۱۴۹ - ----- لیست کامل Capabilities در لینوکس
- ۱۵۳ - ----- مشاهده قابلیت‌های یک فرآیند در لینوکس
- ۱۵۳ - ----- ساختار داده‌های نمایش داده‌شده
- ۱۵۴ - ----- یک مثال از فرآیند متعلق به root
- ۱۵۴ - ----- رمزگشایی مقادیر هگزادسیمال
- ۱۵۴ - ----- مثال کاربردی: بررسی قابلیت‌های ping
- ۱۵۵ - ----- ابزار ساده تر getpcaps

- ۱۵۵ - قابلیت‌های باینری‌ها و کاربران در لینوکس
- ۱۵۶ - قابلیت‌های باینری‌ها
- ۱۵۶ - جست‌وجوی باینری‌های دارای قابلیت
- ۱۵۶ - حذف قابلیت‌ها با capsh
- ۱۵۶ - حذف دائمی قابلیت‌ها از باینری
- ۱۵۷ - قابلیت‌های کاربران (User Capabilities)
- ۱۵۷ - مقدمه‌ای بر USBGuard
- ۱۵۷ - ضرورت حفاظت از سیستم در برابر دستگاه‌های USB
- ۱۵۸ - معماری و مؤلفه‌های اصلی USBGuard
- ۱۵۸ - رابط خط فرمان: ابزار مدیریت برای administrators
- ۱۵۸ - زبان قوانین: بیان سیاست‌های امنیتی به صورت ساختاریافته
- ۱۵۸ - API++ پایه‌ای برای توسعه و یکپارچه‌سازی
- ۱۵۹ - پیاده‌سازی و پیکربندی عملی USBGuard
- ۱۵۹ - ایجاد مجموعه قوانین اولیه: پایه‌ریزی سیاست امنیتی
- ۱۵۹ - سفارشی‌سازی مجموعه قوانین: تنظیم سیاست بر اساس نیازهای خاص
- ۱۵۹ - راه‌اندازی سرویس: USBGuard فعال‌سازی حفاظت بلادرنگ
- ۱۵۹ - مدیریت دستگاه‌های USB در عمل
- ۱۵۹ - فهرست کردن دستگاه‌ها: مشاهده وضعیت فعلی
- ۱۵۹ - مدیریت دسترسی دستگاه‌ها: اعطا و سلب مجوز
- ۱۶۰ - درک تفاوت بین block و reject: دو سطح از حفاظت
- ۱۶۰ - ایجاد لیست سفید و لیست سیاه: استراتژی‌های مختلف کنترل دسترسی
- ۱۶۰ - ملاحظات امنیتی حیاتی: حفاظت از رابط IPC
- ۱۶۱ - مدیریت استثناها و موقعیت‌های خاص
- ۱۶۱ - نظارت و audit پیوسته
- ۱۶۱ - معرفی: ASLR انقلابی در امنیت حافظه
- ۱۶۱ - تهدیدات امنیتی قبل از ظهور ASLR
- ۱۶۱ - معماری و پیاده‌سازی ASLR در لینوکس
- ۱۶۱ - سطوح مختلف تصادفی‌سازی در ASLR
- ۱۶۲ - محدودیت‌ها و چالش‌های ASLR
- ۱۶۲ - مدیریت و پیکربندی ASLR در لینوکس
- ۱۶۲ - پارامتر kernel.randomize_va_space: کنترل مرکزی ASLR
- ۱۶۳ - روش‌های مدیریت پارامتر ASLR
- ۱۶۳ - پیکربندی دائمی از طریق /etc/sysctl.conf:
- ۱۶۳ - روش جایگزین با استفاده از فایل‌های اختصاصی:
- ۱۶۳ - بهینه‌سازی ASLR برای محیط‌های مختلف
- ۱۶۴ - بررسی تاثیر ASLR بر امنیت عملی
- ۱۶۴ - مفهوم بیت NX و معماری آن

- ۱۶۴ - اهمیت بیت NX در مقابله با حملات سایبری
- ۱۶۵ - بررسی فعال بودن بیت NX در سیستم
- ۱۶۵ - Exec-Shield
- ۱۶۶ - امنیت پروتکل ICMP و تنظیمات کرنل
- ۱۶۶ - تهدیدات امنیتی مرتبط با ICMP
- ۱۶۶ - غیرفعال کردن پاسخ به درخواست‌های ICMP
- ۱۶۷ - مدیریت پاسخ به درخواست‌های ICMP Broadcast
- ۱۶۷ - پیکربندی دائمی تنظیمات امنیتی
- ۱۶۸ - احراز هویت مبتنی بر کلید در SSH
- ۱۶۸ - احراز هویت مبتنی بر کلید عمومی
- ۱۶۸ - چالش‌های احراز هویت مبتنی بر کلید عمومی
- ۱۶۹ - مزایای استفاده از مرجع گواهی SSH
- ۱۶۹ - پیاده‌سازی عملی مرجع گواهی SSH
- ۱۷۰ - امضای کلید میزبان سرور
- ۱۷۰ - پیکربندی اجزاء برای استفاده از گواهی‌های میزبان
- ۱۷۰ - ایجاد اعتماد به مرجع گواهی
- ۱۷۰ - تنظیم مرجع گواهی قابل اعتماد
- ۱۷۱ - مفهوم و معماری محیط‌های Chroot
- ۱۷۱ - مزایای محیط Chroot
- ۱۷۲ - فناوری‌های مدرن ایزوله‌سازی
- ۱۷۲ - مجازی‌سازی (Virtualization)
- ۱۷۲ - آسیب‌پذیری‌های سخت‌افزاری Meltdown و Spectre
- ۱۷۲ - آسیب‌پذیری Spectre
- ۱۷۲ - آسیب‌پذیری Meltdown
- ۱۷۳ - بررسی وضعیت patch شدن سیستم
- ۱۷۳ - پیکربندی و امن‌سازی Grub
- ۱۷۳ - معرفی PolKit و نقش آن در امنیت سیستم
- ۱۷۳ - ضرورت امن‌سازی Grub
- ۱۷۳ - امن‌سازی Grub نسخه ۱ و ۲

فصل ۶ - ۱۷۸ - 332.2 تشخیص نفوذ در میزبان (Host Intrusion Detection) - ۱۷۸ -

- ۱۷۹ - نقش حیاتی ابزارهای تشخیص تهدید در امنیت سایبری
- ۱۷۹ - اهمیت درک عمقی مفاهیم امنیتی
- ۱۷۹ - معرفی ابزارهای کلیدی تشخیص تهدید
- ۱۷۹ - AIDE استاندارد طلایی تشخیص نفوذ
- ۱۷۹ - OPENScap - راه‌حل جامع مانیتورینگ از RedHat
- ۱۷۹ - Linux Malware Detect شکارچی تخصصی بدافزارها

- ۱۸۰ - _____ Chkrootkit و Rkhunter متخصصان کشف روتکیت
- ۱۸۰ - _____ بررسی عمیق AIDE
- ۱۸۰ - _____ امن سازی AIDE با SELinux
- ۱۸۰ - _____ نصب و راه اندازی اولیه AIDE
- ۱۸۱ - _____ پیکربندی پیشرفته AIDE
- ۱۸۱ - _____ درک فایل پیکربندی `/etc/aide.conf`
- ۱۸۳ - _____ قوانین از پیش تعریف شده در AIDE
- ۱۸۴ - _____ ایجاد قوانین سفارشی
- ۱۸۴ - _____ پیاده سازی قوانین بر روی فایل ها و دایرکتوری ها
- ۱۸۴ - _____ مدیریت عملیاتی AIDE
- ۱۸۵ - _____ نقش حیاتی قوانین در ردیابی و تشخیص فایل ها
- ۱۸۵ - _____ اهمیت تنظیم قوانین قبل از مقداردهی اولیه پایگاه داده
- ۱۸۶ - _____ قوانین پیش فرض و سفارشی در AIDE
- ۱۸۶ - _____ یکپارچه سازی با استاندارد SCAP
- ۱۸۷ - _____ درک مفهوم OpenSCAP و جایگاه آن در امنیت سایبری
- ۱۸۷ - _____ مؤلفه های کلیدی معماری OpenSCAP
- ۱۸۷ - _____ ویژگی های کلیدی OpenSCAP
- ۱۸۸ - _____ نصب OpenSCAP
- ۱۸۸ - _____ Linux Malware Detect (LMD)
- ۱۸۸ - _____ مکانیزم عملکرد LMD
- ۱۸۹ - _____ نصب و راه اندازی Linux Malware Detect
- ۱۹۰ - _____ آناتومی فایل پیکربندی (`conf.maldet`)
- ۱۹۳ - _____ مفهوم روتکیت و خطرات بنیادین آن
- ۱۹۴ - _____ نصب و راه اندازی chkrootkit در توزیع های مختلف لینوکس
- ۱۹۶ - _____ معرفی Rkhunter
- ۱۹۶ - _____ نصب و به روز رسانی
- ۱۹۷ - _____ ایجاد پایگاه داده پایه: نقش حیاتی `-propupd`
- ۱۹۷ - _____ پیکربندی پیشرفته
- ۱۹۸ - _____ حسابرسی سیستم با Auditd
- ۱۹۸ - _____ تفاوت بنیادین حسابرسی (Auditing) و ثبت رویداد (Logging)
- ۱۹۹ - _____ آناتومی یک رکورد حسابرسی
- ۱۹۹ - _____ معماری سیستم حسابرسی
- ۱۹۹ - _____ نصب و پیکربندی Auditd
- ۲۰۰ - _____ ابزارهای مدیریت و تحلیل لاگ ها
- ۲۰۱ - _____ پیکربندی `pam_tty_audit`
- ۲۰۲ - _____ ارسال لاگ ها به سیستم های راه دور

فصل ۷ ----- ۲۰۵ -
332.3 کنترل منابع (Resource Control) ----- ۲۰۵ -
وزن: ۳ ----- ۲۰۵ -

- ۲۰۶ - مدیریت منابع در لینوکس
- ۲۰۶ - دستور: ulimit: کنترلگر منابع پوسته
- ۲۰۷ - محدودیت‌های نرم (Soft) در مقابل سخت (Hard):
- ۲۰۸ - پیکربندی پایدار با `/etc/security/limits.conf`
- ۲۰۸ - مثال‌های کاربردی
- ۲۰۹ - یکپارچگی با سیستم: PAM معماری زیرساختی
- ۲۱۲ - مقدمه‌ای بر `cgroups`:
- ۲۱۲ - معماری: `cgroups` درک سازوکار داخلی
- ۲۱۲ - ساختار سلسله مراتبی: `cgroups` یک مثال عملی
- ۲۱۳ - مشاهده `cgroups` فعال: از طریق سیستم فایل `proc`
- ۲۱۳ - بررسی از طریق سیستم فایل `/sys/fs/cgroup`
- ۲۱۴ - ابزارهای نظارت و مدیریت: `systemd-cgls` و `systemd-cgtop`
- ۲۱۶ - تفاوت‌های کلیدی بین `cgroups` نسخه ۱ و ۲
- ۲۱۶ - کنترل دسترسی اختیاری (Discretionary Access Control – DAC)
- ۲۱۷ - عناصر اصلی DAC
- ۲۱۷ - ابزارهای مرتبط با DAC در لینوکس
- ۲۱۸ - مدیریت دسترسی با دستور `chmod`
- ۲۱۸ - ساختار کلی دستور `chmod`
- ۲۱۸ - تغییر دسترسی‌ها با کدهای نمادین (Symbolic Mode)
- ۲۱۹ - تغییر دسترسی‌ها با کدهای اکتالی (Octal Mode)
- ۲۲۲ - بیت‌های ویژه در لینوکس: `SetUID`، `SetGID` و `Sticky Bit`
- ۲۲۲ - بیت (Set User ID) `SetUID`
- ۲۲۲ - امنیت `SetUID`
- ۲۲۲ - بیت (Set Group ID) `SetGID`
- ۲۲۳ - بیت `Sticky Bit`
- ۲۲۳ - نحوه تنظیم بیت‌های ویژه
- ۲۲۴ - تفاوت نمایش `s` و `t` و `S` و `T`
- ۲۲۵ - تغییر مالکیت فایل‌ها با دستور `chown`
- ۲۲۵ - ساختار کلی دستور `chown`
- ۲۲۶ - تغییر گروه با دستور `chgrp`
- ۲۲۷ - ویژگی‌های توسعه یافته فایل‌ها (Extended Attributes – xattr)
- ۲۲۷ - Extended Attributes چیست؟
- ۲۲۸ - دستورات مدیریت `xattr`
- ۲۲۸ - انواع `xattr` در namespace

- ۲۳۰ - قرار دادن توضیح روی یک فایل کانفیگ برای مستندسازی
- ۲۳۰ - نمایش توضیحات اضافه شده به فایل کانفیگ
- ۲۳۰ - **مثال**
- ۲۳۰ - قرار دادن نسخه نرم‌افزار روی باینری برای ردیابی
- ۲۳۰ - نمایش نسخه ذخیره‌شده روی باینری
- ۲۳۰ - افزودن نام مالک پروژه روی پوشه سورس کد
- ۲۳۰ - نمایش نام مالک پوشه پروژه
- ۲۳۰ - قرار دادن هش فایل برای بررسی یکپارچگی
- ۲۳۰ - بررسی مقدار هش ذخیره‌شده در attribute
- ۲۳۰ - علامت‌گذاری یک فایل لاگ به عنوان حساس (مثلاً برای audit)
- ۲۳۰ - نمایش وضعیت حساس بودن فایل لاگ
- ۲۳۰ - لیست‌های کنترل دسترسی (Access Control Lists – ACLs)
- ۲۳۱ - ACL چیست؟
- ۲۳۱ - فرمت ACL
- ۲۳۱ - دستورات مدیریت ACL
- ۲۳۳ - مفهوم ماسک (Mask) در ACL
- ۲۳۳ - ACL پیش‌فرض در دایرکتوری‌ها (Default ACLs)
- ۲۳۳ - سناریوهای کاربردی ACL

فصل ۸ - ۲۳۹ - ۲، ۳۳۳ کنترل دسترسی اجباری (Mandatory Access Control) وزن: ۵ - ۲۳۹ -

- ۲۴۰ - بخش اول: درک کنترل دسترسی اجباری (Mandatory Access Control)
- ۲۴۰ - مقایسه DAC و MAC
- ۲۴۱ - چرا MAC امن‌تر است؟
- ۲۴۱ - نکته مهم درباره نقش MAC در Root
- ۲۴۱ - SELinux (Security-Enhanced Linux)
- ۲۴۱ - مفهوم اصلی: Labeling (برچسب‌گذاری)
- ۲۴۲ - AppArmor (Application Armor)
- ۲۴۲ - تفاوت با SELinux
- ۲۴۳ - Smack (Simplified Mandatory Access Control Kernel)
- ۲۴۳ - SELinux
- ۲۴۴ - سیاست‌ها در SELinux
- ۲۴۴ - انواع سیاست‌ها
- ۲۴۴ - برچسب‌گذاری (Labeling)
- ۲۴۵ - حالت‌های اجرایی SELinux
- ۲۴۶ - اجرای نوعی (Type Enforcement)
- ۲۴۶ - ثبت رویدادها و عیب‌یابی
- ۲۴۶ - ابزارها و دستورات SELinux

- ۱. ابزار semanage _____ - ۲۴۷ -
- ۲. ابزار restorecon _____ - ۲۴۷ -
- ۳. ابزار seinfo _____ - ۲۴۷ -
- ۴. دستور setsebool _____ - ۲۴۸ -
- ۵. سایر دستورات پر کاربرد _____ - ۲۴۸ -
- _____ - ۲۴۸ - getenforce
- _____ - ۲۴۸ - setenforce
- _____ - ۲۴۸ - sestatus
- _____ - ۲۴۸ - ausearch
- _____ - ۲۴۸ - sealert
- _____ - ۲۴۹ - سرویس‌ها و نمونه‌های کاربردی SELinux
- _____ - ۲۴۹ - وب‌سرور Apache
- _____ - ۲۴۹ - وب‌سرور Nginx
- _____ - ۲۵۰ - پایگاه داده MySQL / MariaDB
- _____ - ۲۵۰ - سایر سرویس‌ها
- _____ - ۲۵۰ - نوشتن و ویرایش سیاست‌های SELinux
- _____ - ۲۵۰ - ساختار سیاست‌های SELinux
- _____ - ۲۵۱ - ابزارهای کار با سیاست‌ها

فصل ۹ _____ - ۲۵۴ -

۳۳۴٫۱ امن سازی شبکه (Network Hardening) موضوع: ۳۳۴ امنیت شبکه وزن: ۴ _____ - ۲۵۴ -

- _____ - ۲۵۵ - مقدمه و معرفی RADIUS و AAA
- _____ - ۲۵۵ - ۲. مجوزدهی (Authorization)
- _____ - ۲۵۵ - ۳. حسابرسی یا حسابداری (Accounting)
- _____ - ۲۵۵ - اهمیت انعطاف‌پذیری در RADIUS
- _____ - ۲۵۵ - کارایی و گستردگی RADIUS
- _____ - ۲۵۵ - معرفی FreeRADIUS و اهمیت آن
- _____ - ۲۵۶ - کاربردهای FreeRADIUS در دنیای واقعی
- _____ - ۲۵۷ - نصب و راه‌اندازی FreeRADIUS
- _____ - ۲۵۹ - تست اولیه FreeRADIUS
- _____ - ۲۵۹ - ابزار مدیریتی radmin
- _____ - ۲۵۹ - نحوه دسترسی به radmin
- _____ - ۲۶۰ - قابلیت‌های کلیدی radmin
- _____ - ۲۶۱ - ابزارهای تست و اشکال‌زدایی در FreeRADIUS
- _____ - ۲۶۱ - ابزار radtest
- _____ - ۲۶۲ - radclient
- _____ - ۲۶۲ - مقایسه radclient و radtest

- ۲۶۲ - ابزارهای گزارش‌گیری و مانیتورینگ کاربران
- ۲۶۳ - radlast ابزار
- ۲۶۳ - radwho ابزار
- ۲۶۴ - tcpdump ابزار پایه‌ای ضبط و بررسی بسته‌ها
- ۲۶۴ - PCAP: قالب استاندارد ضبط بسته‌ها و فیلترها
- ۲۶۵ - Wireshark: آنالیزگر گرافیکی جامع بسته‌ها
- ۲۶۵ - TShark: نسخه خط فرمانی Wireshark
- ۲۶۶ - ndpmon: پایش IPv6 Neighbor Discovery
- ۲۶۶ - Kismet: چارچوب شنود و شناسایی شبکه‌های بی‌سیم
- ۲۶۶ - Aircrack-ng: مجموعه ابزار برای ارزیابی امنیت Wi-Fi
- ۲۶۷ - Bettercap: چاقوی سوئیچی برای حملات و آنالیز شبکه

فصل ۱۰ - ۲۷۳ -

۳۳۴٫۲ تشخیص نفوذ شبکه (Network Intrusion Detection) وزن: ۴ - ۲۷۳ -

- ۲۷۳ - آشنایی با ntop و ntopng
- ۲۷۳ - ntop چیست؟
- ۲۷۴ - حالت‌های اجرایی ntop
- ۲۷۴ - دستورهای رایج در ntop
- ۲۷۴ - جایگزینی ntop با ntopng
- ۲۷۵ - مقدمه‌ای بر پایش ترافیک شبکه
- ۲۷۵ - لایه ۲ و لایه ۳ در شبکه
- ۲۷۵ - حالت‌های اجرایی ntop
- ۲۷۶ - دستورات مهم ntop
- ۲۷۶ - ntopng نسل جدید ntop
- ۲۷۷ - پایش و مانیتورینگ شبکه با Cacti
- ۲۷۷ - ویژگی‌های اصلی Cacti
- ۲۷۸ - نصب و راه‌اندازی Cacti
- ۲۷۸ - ساختار کاری Cacti
- ۲۷۸ - نمونه کاربردهای Cacti در شبکه
- ۲۷۹ - مقایسه Cacti با ntopng
- ۲۷۹ - Nagios
- ۲۷۹ - ویژگی‌های کلیدی Nagios
- ۲۸۰ - اجزای اصلی Nagios
- ۲۸۰ - نصب و راه‌اندازی Nagios
- ۲۸۱ - Nagios در مقایسه با Cacti و ntopng
- ۲۸۱ - یکپارچه‌سازی Nagios با سایر ابزارها
- ۲۸۱ - نقاط قوت و ضعف Nagios
- ۲۸۲ - Zabbix

۲۸۲	ویژگی های کلیدی Zabbix
۲۸۳	معماری Zabbix
۲۸۳	نصب و راه اندازی Zabbix
۲۸۳	مانیتورینگ در Zabbix
۲۸۴	هشدارها (Triggers)
۲۸۴	داشبوردها و گراف ها
۲۸۴	مقایسه Zabbix با Nagios
۲۸۴	Snort
۲۸۴	معماری Snort
۲۸۵	حالت های کاری Snort
۲۸۵	Rule ها در Snort
۲۸۶	نصب و راه اندازی Snort
۲۸۷	پایگاه Rule ها
۲۸۷	Snort در حالت IPS
۲۸۷	مزایا و معایب Snort
۲۸۷	PCAP
۲۸۸	Wireshark
۲۸۸	فیلترها در Wireshark
۲۸۸	TShark
۲۸۹	Kismet
۲۸۹	Aircrack-ng
۲۸۹	Bettercap
۲۸۹	تهدیدات شبکه ای و مقابله با آن ها
۲۹۰	OpenVAS
۲۹۰	ویژگی های کلیدی OpenVAS
۲۹۰	تاریخچه و توسعه OpenVAS
۲۹۰	نصب و پیکربندی OpenVAS
۲۹۰	به روز رسانی Feed ها
۲۹۱	نقش NASL در OpenVAS
۲۹۱	Network Vulnerability Tests (NVTs)
۲۹۱	فایل پیکربندی openvassd.conf
۲۹۲	امنیت در OpenVAS
۲۹۵	فصل ۱۱
۲۹۵	فایروال و فیلترینگ بسته ها
۲۹۶	Netfilter
۲۹۶	وظایف اصلی Netfilter

۲۹۶	اهمیت Netfilter در امنیت
۲۹۶	آشنایی با iptables
۲۹۶	مزایای iptables
۲۹۶	معایب iptables
۲۹۶	جداول (Tables) و زنجیره‌ها (Chains) در iptables
۲۹۷	اهداف (Targets) و سیاست‌ها (Policies) در iptables
۲۹۷	اهداف اصلی در iptables
۲۹۷	سیاست پیش‌فرض (Default Policy)
۲۹۸	مدیریت قوانین iptables
۲۹۸	دستورهای اصلی iptables
۲۹۸	ذخیره و بازیابی قوانین iptables
۲۹۸	ذخیره‌سازی قوانین
۲۹۹	بازیابی قوانین
۲۹۹	فیلترینگ IPv6 با iptables
۳۰۰	مفاهیم پیشرفته فایروال
۳۰۰	IP Sets
۳۰۰	دستورات اصلی ipset
۳۰۰	پایداری IP sets
۳۰۱	شبکه DMZ
۳۰۱	Connection Tracking
۳۰۱	نقش conntrack
۳۰۱	ابزارهای conntrack
۳۰۱	NAT (Network Address Translation)
۳۰۲	انواع NAT
۳۰۲	ebtables
۳۰۲	nftables

فصل ۱۲ - ۳۰۴

۳۳۴٫۴ شبکه‌های خصوصی مجازی (Virtual Private Networks) وزن: ۴ - ۳۰۴

۳۰۵	آشنایی با OpenVPN
۳۰۵	نحوه‌ی کار OpenVPN
۳۰۵	نصب OpenVPN
۳۰۵	ایجاد گواهی‌نامه با Easy-RSA
۳۰۶	ایجاد گواهی سرور و کلاینت
۳۰۶	بیکربندی سرور OpenVPN
۳۰۷	بیکربندی کلاینت
۳۰۷	تنظیمات امنیتی و فایروال
۳۰۷	آشنایی با IPsec

- ۳۰۷ - نحوه‌ی عملکرد IPsec
- ۳۰۸ - حالت‌های کاری IPsec
- ۳۰۸ - کاربردهای IPsec
- ۳۰۸ - نصب و راه‌اندازی IPsec در لینوکس
- ۳۰۹ - مدیریت سرویس IPsec
- ۳۰۹ - مزایا و معایب IPsec
- ۳۰۹ - strongSwan
- ۳۰۹ - ویژگی‌های strongSwan
- ۳۱۰ - نصب strongSwan در لینوکس
- ۳۱۱ - مزایای strongSwan نسبت به دیگر پیاده‌سازی‌ها
- ۳۱۱ - WireGuard
- ۳۱۱ - نصب WireGuard
- ۳۱۲ - پیکربندی WireGuard
- ۳۱۲ - ابزار wg
- ۳۱۲ - ایجاد فایل پیکربندی wg0.conf
- ۳۱۳ - فعال‌سازی IP Forwarding
- ۳۱۳ - پیکربندی کلاینت WireGuard
- ۳۱۴ - افزودن کلاینت به سرور

فصل ۱۳ - ۳۱۷ -

۱۳/۳۳۵ آسیب‌پذیری‌ها و تهدیدات امنیتی رایج موضوع: ۳۳۵ ارزیابی تهدیدات و آسیب‌پذیری‌ها وزن: ۲-۳۱۷ -

- ۳۱۷ - فهرست جزئی از فایل‌ها، اصطلاحات و ابزارهای مورد استفاده:
- ۳۱۸ - ویروس‌های رایانه‌ای
- ۳۱۸ - تفاوت ویروس و کرم رایانه‌ای
- ۳۱۸ - بدافزار (Malware)
- ۳۱۸ - تفاوت بدافزار و ویروس
- ۳۱۹ - تهدیدات رایانه‌ای (Computer Threats)
- ۳۱۹ - انواع دیگر بدافزار
- ۳۱۹ - کرم‌ها (Worms)
- ۳۱۹ - روت‌کیت‌ها (Rootkits)
- ۳۲۰ - تروجان‌ها (Trojans)
- ۳۲۰ - تعریف و نحوه‌ی عملکرد تروجان‌ها
- ۳۲۰ - ریشه‌ی نام تروجان
- ۳۲۰ - انواع تروجان‌ها
- ۳۲۰ - کی لاگرها (Keyloggers)
- ۳۲۱ - انواع کی لاگرها
- ۳۲۱ - ۲. کی لاگر سخت‌افزاری
- ۳۲۱ - مهندسی اجتماعی (Social Engineering)

- ۳۲۱ - مهندسی اجتماعی در علوم اجتماعی و امنیت سایبری
- ۳۲۱ - نمونه‌های حملات مهندسی اجتماعی
- ۳۲۲ - حملات Brute Force نیروی بی‌رحمانه
- ۳۲۳ - حملات Rainbow Table جدول رنگین‌کمانی
- ۳۲۳ - سرریز بافر (Buffer Overflow)
- ۳۲۴ - اسکریپت‌نویسی میان‌وبگاهی (Cross-Site Scripting - XSS)
- ۳۲۴ - جعل درخواست میان‌وبگاهی (Cross-Site Request Forgery - CSRF)
- ۳۲۴ - جعل IP (IP Spoofing)
- ۳۲۵ - حملات DDoS (Distributed Denial of Service)
- ۳۲۵ - حملات مرد میانی (Man-in-the-Middle)
- ۳۲۵ - ارتقای دسترسی (Privilege Escalation)
- ۳۲۵ - تزریق SQL (SQL Injection)
- ۳۲۵ - DHCP جعلی (Rogue DHCP)
- ۳۲۵ - نقاط دسترسی جعلی (Rogue Access Point)
- ۳۲۵ - جعل ARP و NDP

فصل ۱۴ - ۳۲۸ -

۳۳۵٫۲ تست نفوذ (Penetration Testing) وزن: ۳ - ۳۲۸ -

- ۳۲۹ - انواع تست نفوذ
- ۳۳۰ - مراحل (فازها)ی تست نفوذ
- ۳۳۰ - شناسایی (Reconnaissance)
- ۳۳۰ - اسکن (Scanning)
- ۳۳۰ - دستیابی به دسترسی (Gaining Access)
- ۳۳۰ - حفظ دسترسی (Maintaining Access)
- ۳۳۱ - پاک‌سازی ردپا (Covering Tracks)
- ۳۳۱ - ابزارها و تکنیک‌ها
- ۳۳۱ - نحوه برنامه‌ریزی و اجرای یک تست نفوذ
- ۳۳۱ - گزارش‌دهی و ارائه نتایج
- ۳۳۲ - مزایا و محدودیت‌ها
- ۳۳۲ - مسائل قانونی و اخلاقی
- ۳۳۲ - بهترین شیوه‌ها (Best Practices)
- ۳۳۲ - نمونه‌های کاربردی و سناریوها
- ۳۳۳ - مزایای تست نفوذ
- ۳۳۳ - ابزارهای محبوب تست نفوذ
- ۳۳۳ - Nmap
- ۳۳۴ - Metasploit
- ۳۳۴ - Wireshark

- ۳۳۴ - _____ Burp Suite
- ۳۳۴ - _____ انتخاب ابزار مناسب و ترکیب آنها
- ۳۳۵ - _____ محدودیت‌ها و نکات احتیاطی در استفاده از ابزارها
- ۳۳۵ - _____ چگونگی ادغام پنتست در برنامه امنیتی سازمان
- ۳۳۵ - _____ خلاصه و راهنمای گام‌به‌گام برای استفاده از تست نفوذ
- ۳۳۵ - _____ هکر اخلاقی دارای مدرک (Certified Ethical Hacker – CEH)
- ۳۳۵ - _____ محتوای آموزشی CEH
- ۳۳۶ - _____ ابزار Nmap و نقش آن در امنیت سایبری
- ۳۳۷ - _____ کاربردهای اصلی Nmap
- ۳۳۷ - _____ انواع روش‌های انتخاب هدف در Nmap
- ۳۳۸ - _____ اسکن پورت‌ها با Nmap
- ۳۳۸ - _____ روش‌های مختلف اسکن در Nmap
- ۳۳۹ - _____ موتور اسکریپت‌نویسی (NSE) Nmap
- ۳۳۹ - _____ چارچوب متاسپلویت (Metasploit Framework)
- ۳۳۹ - _____ معماری Metasploit Framework
- ۳۴۰ - _____ کاربردهای متاسپلویت
- ۳۴۰ - _____ نسخه‌های Metasploit
- ۳۴۱ - _____ ابزار (Social-Engineer Toolkit (SET)
- ۳۴۱ - _____ ویژگی‌ها و قابلیت‌های کلیدی SET

مقدمه

به کتاب LPIC-3 خوش آمدید. این کتاب با هدف راهنمایی شما برای آزمون LPIC-3 نوشته شده است. آزمونی که دامنه گسترده‌ای از مطالب را پوشش می‌دهد و موفقیت در آن نیازمند تجربه عملی و درک عمیق مفاهیم مدیریت سیستم‌های لینوکس است. ایده نگارش این کتاب زمانی به ذهن من رسید که در حال به‌روزرسانی مدرک خود بودم و تصمیم گرفتم برخی از نکات و تجربیات خود را با شما به اشتراک بگذارم.

شما می‌توانید این کتاب را به صورت موضوعی مطالعه کنید یا آن را به عنوان یک منبع برای آماده‌سازی آزمون استفاده کنید، اما فراموش نکنید که گذراندن LPIC-1 و LPIC-2 پیش‌نیازهای این کتاب هستند و بدون داشتن دانش پایه‌ای که در این دو سطح ارائه شده است، مطالعه LPIC-3 دشوار خواهد بود.

شاخه‌های مختلف LPIC-3

۱. LPIC-3 Security (303)

- تمرکز: امنیت پیشرفته لینوکس
- مباحث: رمزنگاری، VPN، مدیریت کلید و گواهی (PKI)، امنیت شبکه، کنترل دسترسی (PAM, ACL)، امنیت سرویس‌ها (SSH, Squid, Apache) و ابزارهایی مثل SELinux و AppArmor.
- برای مدیران سیستم که می‌خوان سیاست‌های امنیتی قوی در سازمان پیاده کنند.

۲. LPIC-3 Mixed Environment (300)

- تمرکز: یکپارچگی لینوکس با سایر سیستم‌ها مثل ویندوز و دایرکتوری‌های LDAP.
- مباحث: Samba و یکپارچه‌سازی با Active Directory، سرورهای فایل و پرینت مشترک، Kerberos و LDAP، مهاجرت کاربران بین ویندوز و لینوکس.
- مناسب برای مدیرانی که شبکه‌های هیبریدی (Mixed Environment) رو مدیریت می‌کنن.

۳. LPIC-3 Virtualization and High Availability (304)

- تمرکز: مجازی‌سازی و در دسترس‌پذیری بالا (HA).
- مباحث: Xen، KVM، مدیریت ماشین‌های مجازی، خوشه‌بندی (Clustering)، load balancing، SAN/NAS، ابزارهای HA مثل Pacemaker و Corosync.
- مناسب برای کسانی که با دیتاستر، Cloud یا زیرساخت‌های Enterprise کار می‌کنن.

۴. LPIC-3 Enterprise Messaging and Collaboration (306)

- تمرکز: سرویس‌های ایمیل و همکاری سازمانی.
- مباحث: Postfix، Dovecot، Cyrus، SpamAssassin، Amavis، امنیت ایمیل (TLS, DKIM, SPF)، ابزارهای groupware (مثل Kolab، Zarafa).
- مناسب برای مدیران شبکه‌هایی که سرویس ایمیل سازمانی رو روی لینوکس پیاده‌سازی می‌کنن.

LPIC-3 شاخه‌های مختلفی دارد و در این کتاب ما بر روی شاخه امنیت یعنی ۳۰۳ (Security) تمرکز کرده‌ایم. امنیت سیستم‌های لینوکس یکی از مهم‌ترین مباحث در حوزه فناوری اطلاعات و مدیریت شبکه است، زیرا هر روز تهدیدات جدیدی علیه سیستم‌ها و داده‌ها شکل می‌گیرد. آزمون امنیت LPIC-3 به شما کمک می‌کند تا دانش و مهارت‌های خود را در زمینه شناسایی تهدیدات، پیاده‌سازی سیاست‌های امنیتی و حفاظت از اطلاعات سازمان‌ها بهبود بخشید.

برای کسانی که با LPIC آشنایی ندارند، LPIC مخفف Linux Professional Institute Certification است، یک سازمان بین‌المللی که مدارک معتبر و شناخته‌شده‌ای در حوزه لینوکس ارائه می‌دهد. مدرک LPIC-3، سطح سوم این مسیر است و برای مدیران سیستم پیشرفته طراحی شده است که مسئولیت‌های پیچیده امنیت، مجازی‌سازی و مدیریت شبکه را بر عهده دارند.

آزمون امنیت LPIC-3 شامل یک امتحان اصلی با کد 303 است. این امتحان به گونه‌ای طراحی شده که توانایی شما را در مدیریت امنیت سیستم‌های لینوکس ارزیابی کند و شما را برای مواجهه با چالش‌های واقعی در محیط‌های عملیاتی آماده سازد. در ادامه، ما به بررسی جزئیات این آزمون و نحوه آماده‌سازی برای آن خواهیم پرداخت.

چرا LPIC-3 Security اهمیت دارد؟

امنیت اطلاعات یکی از حیاتی‌ترین بخش‌های هر سازمان و شبکه‌ای است. با گسترش استفاده از اینترنت و سیستم‌های ابری، حملات سایبری روز به روز پیچیده‌تر و رایج‌تر می‌شوند. بنابراین، داشتن دانش عملی و تخصصی در زمینه امنیت لینوکس نه تنها برای مدیران سیستم بلکه برای هر حرفه‌ای که با فناوری اطلاعات سر و کار دارد، ضروری است.

مدرک LPIC-3 Security به شما نشان می‌دهد که توانایی شناسایی نقاط ضعف، اجرای سیاست‌های امنیتی، مدیریت دسترسی کاربران، رمزنگاری داده‌ها و پیاده‌سازی راهکارهای محافظتی پیشرفته را دارید. این مهارت‌ها باعث می‌شود که شما بتوانید به عنوان یک متخصص امنیت سیستم‌های لینوکس شناخته شوید و در سازمان‌های بزرگ یا پروژه‌های پیچیده ایفای نقش کنید.

یکی از نکات مهم در آزمون LPIC-3 Security، تمرکز آن بر تجربه عملی است. برخلاف سطح‌های قبلی که بیشتر به مفاهیم پایه و دانش نظری پرداخته‌اند، در این سطح، شما باید بتوانید دستورات، تنظیمات و ابزارهای امنیتی را به طور عملی اجرا کنید و تحلیل کنید. بنابراین، مطالعه صرفاً کتاب یا جزوه کافی نیست و شما نیاز دارید که زمان قابل توجهی را در محیط واقعی یا شبیه‌سازی شده لینوکس صرف تمرین و تجربه عملی کنید.

ساختار کتاب و نحوه مطالعه آن

این کتاب به گونه‌ای طراحی شده است که شما بتوانید آن را به دو روش مطالعه کنید:

۱. **مطالعه موضوعی:** در این روش شما فصل‌ها و بخش‌های مختلف را بر اساس موضوع مطالعه می‌کنید. مثلاً می‌توانید ابتدا بخش مدیریت

دسترسی کاربران را بخوانید، سپس به بخش رمزنگاری داده‌ها بروید و در نهایت سیاست‌های امنیتی شبکه را بررسی کنید. این روش برای کسانی مناسب است که می‌خواهند دانش خود را به صورت عمیق و جامع توسعه دهند.

۲. **مطالعه برای آزمون:** اگر هدف اصلی شما قبولی در آزمون است، می‌توانید کتاب را به گونه‌ای مطالعه کنید که ابتدا مباحثی که بیشترین

اهمیت در آزمون دارند را مرور کرده و سپس به سراغ جزئیات بروید. در این روش، تمرکز بر روی نکات کلیدی، نمونه سوالات احتمالی و تکنیک‌های عملی است که به شما کمک می‌کند در زمان محدود، بهترین نتیجه را کسب کنید.

فراموش نکنید که این کتاب پیش‌نیازهای LPIC-1 و LPIC-2 را مفروض گرفته است. بنابراین، اگر شما با مفاهیم پایه لینوکس، مدیریت کاربران و گروه‌ها، دستورات سیستم، شبکه و امنیت اولیه آشنا نیستید، مطالعه این کتاب ممکن است دشوار باشد. توصیه می‌کنم در ابتدا مرور سریعی بر مباحث LPIC-1 و LPIC-2 داشته باشید تا پایه محکم و قابل اطمینانی برای یادگیری مفاهیم پیشرفته امنیت LPIC-3 ایجاد کنید.

مباحث اصلی آزمون LPIC-3 Security

آزمون ۳۰۳ یا همان LPIC-3 Security شامل چند حوزه کلیدی است که هر کدام نیازمند درک عمیق و تجربه عملی است. در ادامه به مهم‌ترین حوزه‌ها اشاره می‌کنیم و توضیحات بیشتری ارائه می‌دهیم:

۱. مدیریت دسترسی و احراز هویت:

یکی از مهم‌ترین مباحث امنیتی، کنترل دسترسی کاربران و مدیریت احراز هویت است. در این بخش شما باید بتوانید کاربران و گروه‌ها را مدیریت کنید، سطوح دسترسی را تنظیم نمایید، و از ابزارهای امنیتی مانند PAM و LDAP برای احراز هویت امن استفاده کنید.

۲. رمزنگاری و امنیت داده‌ها:

حفاظت از داده‌ها با استفاده از تکنیک‌های رمزنگاری یکی از اصلی‌ترین وظایف مدیران امنیت سیستم است. شما باید با مفاهیمی مانند SSL/TLS، رمزگذاری فایل‌ها، مدیریت کلیدها و گواهی‌های دیجیتال آشنا باشید و توانایی پیاده‌سازی آنها را در محیط لینوکس داشته باشید.

۳. مدیریت امنیت شبکه:

شبکه یکی از مهم‌ترین نقاط آسیب‌پذیری در هر سازمان است. در این بخش، شما باید بتوانید سیاست‌های امنیتی شبکه را پیاده‌سازی کنید، دیوارهای آتش (firewall) را تنظیم کنید، و از ابزارهای مانیتورینگ برای شناسایی تهدیدات شبکه‌ای استفاده کنید.

۴. شناسایی تهدیدات و پاسخ به حوادث:

امنیت تنها به پیشگیری محدود نمی‌شود، بلکه توانایی پاسخ سریع و مناسب به حملات و تهدیدات نیز بسیار اهمیت دارد. در این بخش، شما با تکنیک‌های شناسایی تهدیدات، مانیتورینگ سیستم، تحلیل لاگ‌ها و پاسخ به حوادث امنیتی آشنا می‌شوید.

۵. مدیریت آسیب‌پذیری‌ها و بروزرسانی سیستم:

سیستم‌های لینوکس نیز مانند هر نرم‌افزار دیگری ممکن است آسیب‌پذیری داشته باشند. شما باید توانایی ارزیابی نقاط ضعف سیستم، نصب بروزرسانی‌ها و پیچ‌های امنیتی، و مدیریت ریسک‌ها را داشته باشید.

اگر بخواهیم کمی عمیق‌تر به این مباحث بپردازیم، باید بگوییم که LPIC-3 Security نه تنها شما را برای مدیریت یک سیستم امن آماده می‌کند، بلکه مهارت‌های تحلیلی و تفکر انتقادی شما را نیز تقویت می‌کند. مثلاً وقتی با یک حمله سایبری مواجه می‌شوید، باید بتوانید علت آن را تحلیل کنید، راهکار مناسب را انتخاب کنید و از بروز مشکل مشابه در آینده جلوگیری کنید.

علاوه بر این، در این کتاب سعی شده است که هر مفهوم همراه با مثال‌های عملی و دستورات واقعی لینوکس ارائه شود. این روش باعث می‌شود که مطالعه کتاب نه تنها جنبه نظری داشته باشد، بلکه شما بتوانید مهارت‌های خود را به صورت عملی تمرین کنید و در محیط واقعی یا شبیه‌سازی شده تجربه کسب نمایید.

فصل ۱

۳۳۱،۱ گواهی‌نامه‌های X.509 و زیرساخت کلید عمومی (PKI)

موضوع ۳۳۱: رمزنگاری

وزن در آزمون: ۵٪

توضیح کلی

در این بخش، داوطلبان باید درک کاملی از گواهی‌نامه‌های X.509 و زیرساخت کلید عمومی (Public Key Infrastructure) یا (PKI) داشته باشند. همچنین باید بتوانند با استفاده از ابزار **OpenSSL**، یک مرجع صدور گواهی‌نامه (Certification Authority) یا (CA) را پیکربندی و اجرا کرده و گواهی‌های SSL را برای اهداف مختلف صادر کنند. گواهی‌نامه‌های دیجیتال و PKI پایه و اساس امنیت ارتباطات اینترنتی هستند. این گواهی‌نامه‌ها تضمین می‌کنند که اطلاعات بین سرورها و کاربران بدون تغییر و در محیطی امن منتقل می‌شوند. بدون PKI، وبسایت‌ها و سرویس‌ها نمی‌توانند امنیت لازم برای تراکنش‌های حساس مانند بانکداری آنلاین، تجارت الکترونیک یا انتقال داده‌های محرمانه را فراهم کنند.

حوزه‌های اصلی دانش

برای موفقیت در این بخش، شما باید بر موارد زیر تسلط داشته باشید:

۱. درک گواهی‌نامه‌های X.509 و چرخه عمر آنها

- آشنایی با ساختار گواهی‌نامه‌های X.509، شامل فیلدهای اصلی و افزونه‌های X.509v3
- درک فرآیند صدور، تمدید، استفاده و لغو گواهی‌نامه‌ها

۲. زنجیره اعتماد و زیرساخت کلید عمومی (PKI)

- فهم مفهوم زنجیره اعتماد (Trust Chain) و نحوه ایجاد آن بین سرورها و CAها
- آشنایی با شفافیت گواهی‌نامه‌ها (Certificate Transparency) و اهمیت آن در امنیت اینترنت

۳. مدیریت کلیدهای عمومی و خصوصی

- تولید و مدیریت کلیدهای عمومی و خصوصی به صورت امن
- نحوه نگهداری و حفاظت از کلیدها برای جلوگیری از دسترسی غیرمجاز

۴. ایجاد و مدیریت مرجع صدور گواهی‌نامه (CA)

- پیکربندی و راه‌اندازی CA
- ایمن‌سازی و مدیریت فرآیند صدور گواهی برای سرورها و کلاینت‌ها

۵. صدور و مدیریت گواهی‌نامه‌ها

- درخواست (CSR)، امضا و مدیریت گواهی‌های سرور و کلاینت
- لغو گواهی‌نامه‌ها و مدیریت CAهای لغوشده

۶. آشنایی اولیه با ابزارها و سرویس‌های صدور خودکار گواهی‌نامه

- Let's Encrypt، ACME و Certbot برای صدور خودکار گواهی‌های SSL
- CFSSL (CloudFlare SSL) و کاربردهای اولیه آن

الگوریتم (Algorithm)

عنصر دوم رمزنگاری، الگوریتم است. الگوریتم در واقع مجموعه‌ای از قوانین و فرمول‌های ریاضی است که تعیین می‌کند چگونه داده‌های ساده – Plaintext به داده‌های رمز شده – Ciphertext – تبدیل شوند و برعکس. الگوریتم‌ها روش‌های گوناگونی دارند و هر کدام از آن‌ها دارای نقاط قوت و ضعف خاص خود هستند.

اگر کلید را همان قفل بدانیم، الگوریتم نقش مکانیزم داخلی قفل را دارد. قفل ممکن است ساده یا پیچیده طراحی شده باشد. هر چه این مکانیزم پیچیده‌تر باشد، باز کردن قفل بدون کلید صحیح دشوارتر خواهد بود.

الگوریتم‌های رمزنگاری به دو دسته اصلی تقسیم می‌شوند:

- **الگوریتم‌های متقارن** – مانند AES و Blowfish که در آن‌ها یک کلید مشترک برای رمزگذاری و رمزگشایی استفاده می‌شود.
- **الگوریتم‌های نامتقارن** – مانند RSA که در آن از دو کلید متفاوت ولی مرتبط ریاضی استفاده می‌شود.

برخی الگوریتم‌ها عمومی هستند – یعنی روش کار آن‌ها به صورت کامل منتشر شده و توسط پژوهشگران مورد بررسی قرار گرفته‌اند – و برخی دیگر محرمانه باقی می‌مانند. تجربه نشان داده است که الگوریتم‌های عمومی به مراتب امن‌تر هستند، زیرا در برابر سال‌ها آزمایش، تحلیل و حملات متعدد مقاوم مانده‌اند.

از نمونه‌های مهم الگوریتم‌های رمزنگاری می‌توان به موارد زیر اشاره کرد:

- **AES – Advanced Encryption Standard**: یکی از پرکاربردترین و امن‌ترین الگوریتم‌های رمزنگاری متقارن است که امروزه در اکثر نرم‌افزارها و سخت‌افزارهای امنیتی مورد استفاده قرار می‌گیرد.
- **Blowfish**: الگوریتمی سبک و سریع است که در بسیاری از سیستم‌های قدیمی به کار می‌رفت و هنوز هم در برخی کاربردها دیده می‌شود.
- **3DES – Triple Data Encryption Standard**: نسخه قدیمی‌تر الگوریتم DES است که سه بار متوالی عملیات رمزگذاری را انجام می‌دهد تا امنیت بیشتری ایجاد کند، هرچند امروزه به دلیل سرعت کم و آسیب‌پذیری‌های کشف شده، کمتر مورد استفاده قرار می‌گیرد.

متن ساده و متن رمز شده

در علوم کامپیوتر و به طور کلی در حوزه امنیت اطلاعات، داده‌ها می‌توانند در یکی از دو حالت وجود داشته باشند: **متن ساده و متن رمز شده**. متن ساده – Plaintext – همان داده خام و خوانا است که افراد می‌توانند آن را بدون نیاز به هیچ ابزار خاصی مطالعه کنند. برای مثال وقتی شما یک ایمیل به زبان فارسی یا انگلیسی می‌نویسید، قبل از رمزگذاری، محتوای آن به شکل کاملاً قابل فهم روی صفحه نمایش ظاهر می‌شود. این داده‌ها در صورت انتقال مستقیم از طریق شبکه، برای هر فردی که به مسیر ارتباطی دسترسی داشته باشد قابل مشاهده و سرقت هستند.

در مقابل، **متن رمز شده – Ciphertext** – حالتی است که داده‌ها توسط یک الگوریتم رمزنگاری و با استفاده از یک کلید به شکل غیرقابل خواندن تبدیل می‌شوند. متن رمز شده برای چشم انسان یا حتی برنامه‌های عادی کامپیوتر بی‌معنی به نظر می‌رسد و تنها با داشتن کلید صحیح می‌توان

مرجع صدور گواهی (Certificate Authority – CA) چیست؟

در دنیای اینترنت و ارتباطات دیجیتال، یکی از مهم‌ترین چالش‌ها این است که کاربران بتوانند مطمئن شوند طرف مقابل واقعاً همان کسی است که ادعا می‌کند. وقتی شما به یک وبسایت بانکی متصل می‌شوید و قصد دارید رمز عبور یا اطلاعات کارت بانکی خود را وارد کنید، باید اطمینان داشته باشید که وبسایت واقعاً متعلق به بانک است و یک وبسایت جعلی یا فیشینگ نیست. برای حل این مشکل، مفهومی به نام **مرجع صدور گواهی Certificate Authority – یا به اختصار CA** به وجود آمده است.

مرجع صدور گواهی در حقیقت یک سازمان، شرکت یا نهاد معتبر است که وظیفه آن **تأیید هویت** وبسایت‌ها، افراد یا شرکت‌ها و صدور یک مدرک دیجیتال به نام **گواهی دیجیتال – Digital Certificate** است. این گواهی مانند یک سند رسمی عمل می‌کند که نشان می‌دهد هویت آن وبسایت یا سازمان توسط یک نهاد قابل اعتماد بررسی و تأیید شده است.

به بیان ساده‌تر، CA در اینترنت نقشی مشابه یک **دفتر صدور پاسپورت** در دنیای واقعی دارد. همان‌طور که وقتی پاسپورت می‌گیرید باید هویت خود را اثبات کنید، برای دریافت گواهی دیجیتال نیز یک وبسایت یا سازمان باید مدارک معتبر ارائه دهد تا CA مطمئن شود هویت ادعا شده درست است. نتیجه این فرآیند، صدور یک گواهی دیجیتال است که ارتباط میان هویت واقعی و کلیدهای رمزنگاری را تضمین می‌کند.

چرا CA اهمیت دارد؟

امنیت اینترنت بدون وجود مراجع صدور گواهی تقریباً غیرممکن بود. دلیل آن این است که بخش بزرگی از ارتباطات اینترنتی بر پایه پروتکل **SSL/TLS** انجام می‌شود که همان قفل سبز یا نماد **HTTPS** در مرورگرها است. این پروتکل تضمین می‌کند که ارتباط شما با وبسایت امن و رمزگذاری شده است.

اما سؤال اصلی این است: از کجا بدانیم که وبسایتی که گواهی SSL دارد واقعاً همان سایت اصلی است و نه یک سایت جعلی؟ پاسخ در نقش CA نهفته است. مرورگرها و سیستم‌های عامل به فهرستی از مراجع صدور گواهی معتبر اعتماد می‌کنند. وقتی شما به یک سایت مراجعه می‌کنید، مرورگر گواهی آن سایت را بررسی می‌کند. اگر این گواهی توسط یک CA معتبر صادر شده باشد و اطلاعات آن درست باشد، مرورگر به شما نشان می‌دهد که سایت امن است.

فرآیند صدور گواهی توسط CA

صدور گواهی دیجیتال توسط CA یک فرآیند دقیق و چندمرحله‌ای است. این مراحل بسته به نوع گواهی و سطح اعتماد متفاوت است، اما به طور کلی شامل موارد زیر می‌شود:

۱. **درخواست گواهی: Certificate Request** – وبسایت یا سازمان ابتدا یک جفت کلید عمومی و خصوصی ایجاد می‌کند و سپس درخواستی برای گواهی (**CSR – Certificate Signing Request**) به CA ارسال می‌کند.
۲. **بررسی هویت CA: Identity Verification** – باید مطمئن شود که درخواست‌دهنده واقعاً همان فرد یا سازمانی است که ادعا می‌کند. این بررسی می‌تواند ساده یا بسیار پیچیده باشد. مثلاً در گواهی‌های پایه‌ای تنها مالکیت دامنه بررسی می‌شود، اما در گواهی‌های پیشرفته‌تر ممکن است مدارک ثبتی شرکت یا حتی تماس تلفنی نیز مورد نیاز باشد.
۳. **صدور گواهی: Certificate Issuance** – پس از تأیید هویت، CA یک گواهی دیجیتال صادر می‌کند که شامل اطلاعات هویتی (مانند نام سازمان یا آدرس وبسایت) و کلید عمومی مربوط به آن است.

۳. حملات به الگوریتم‌های رمزنگاری ضعیف

نسخه‌های قدیمی TLS یا SSL ممکن است از الگوریتم‌های رمزنگاری قدیمی مانند MD5 یا RC4 استفاده کنند که امروزه ناامن محسوب می‌شوند. انتخاب الگوریتم‌های قوی و بروز، بخش مهمی از پیکربندی امنیتی سرورها است.

۴. حملات مربوط به گواهی‌های جعلی یا باطل شده

اگر گواهی سرور جعلی باشد یا کلید خصوصی افشا شود، مهاجم می‌تواند به عنوان سرور واقعی ظاهر شود. استفاده از OCSP، CRL و گواهی‌های صادر شده توسط CA معتبر از این تهدید جلوگیری می‌کند.

پیاده‌سازی عملی TLS و HTTPS در Apache HTTPD با استفاده از گواهی‌های X.509

برای ایجاد یک سرویس HTTPS امن Secure HTTPS Service - در Apache HTTPD 2.4 یا بالاتر، باید چند مرحله اصلی را به دقت انجام داد. در این بخش، گام به گام این فرآیند شرح داده می‌شود:

۱. ایجاد کلید خصوصی و گواهی سرور

ابتدا باید یک کلید خصوصی Private Key - ایجاد شود. این کلید برای رمزنگاری ارتباطات و امضای گواهی سرور استفاده می‌شود. دستور زیر نمونه‌ای برای ایجاد یک کلید RSA با طول ۲۰۴۸ بیت و محافظت با رمز عبور است:

```
openssl genrsa -aes128 -out server.key 2048
```

در این دستور:

- genrsa برای تولید کلید RSA استفاده می‌شود.
- aes128 مشخص می‌کند که کلید با الگوریتم AES128 رمزگذاری شود.
- server.key نام فایل کلید خصوصی است.
- 2048 طول کلید بر حسب بیت است.

پس از تولید کلید خصوصی، باید یک گواهی سرور خودامضا Self-Signed Server Certificate - یا یک CSR -

Certificate Signing Request ایجاد کنیم تا توسط CA امضا شود. نمونه دستور برای ایجاد گواهی خودامضا:

```
openssl req -new -x509 -key server.key -out server.crt -days 365
```

در این دستور:

- req برای ایجاد درخواست گواهی استفاده می‌شود.
- x509 مشخص می‌کند که گواهی خودامضا ایجاد شود.
- days 365 مدت اعتبار گواهی را تعیین می‌کند.

۲. پیکربندی Apache HTTPD با mod_ssl

پس از تولید کلید و گواهی، باید Apache را برای استفاده از HTTPS و TLS پیکربندی کنیم. ابتدا اطمینان حاصل کنید که ماژول

mod_ssl فعال است:

```
sudo a2enmod ssl # در سیستم‌های مبتنی بر Debian/Ubuntu
```

سپس فایل پیکربندی httpd.conf یا فایل مربوط به سایت (مثلاً /etc/httpd/sites-available/example.conf) را ویرایش کنید و مسیر کلید خصوصی و گواهی سرور را مشخص کنید:

```
<VirtualHost *:443>
```

احراز هویت کلاینت با گواهی دیجیتال

در مدل کلاسیک HTTPS، کلاینت (کاربر) تنها به بررسی گواهی سرور بسنده می‌کند. مرورگر وب یا نرم‌افزار کلاینت بررسی می‌کند که گواهی ارائه‌شده توسط سرور معتبر بوده، توسط یک مرجع قابل اعتماد (CA) امضا شده و هنوز منقضی نشده است. به این ترتیب، کلاینت به هویت سرور اعتماد کرده و ارتباط رمزگذاری‌شده برقرار می‌شود.

اما در برخی سناریوهای سازمانی، کافی نیست که فقط سرور احراز هویت شود. سرور نیز نیاز دارد مطمئن شود که کلاینتی که قصد برقراری ارتباط دارد واقعاً همان کلاینت مورد انتظار است و نه یک مهاجم. در این شرایط از **Client Certificate Authentication** یا همان احراز هویت کلاینت بر پایه گواهی دیجیتال استفاده می‌شود. در این روش، علاوه بر اینکه کلاینت اعتبار سرور را بررسی می‌کند، سرور نیز اعتبار کلاینت را از طریق گواهی دیجیتال صادرشده توسط یک CA معتبر می‌سنجد.

پیکربندی Apache برای Client Certificate Authentication

برای فعال‌سازی احراز هویت کلاینت در Apache، باید چند دستور کلیدی در پیکربندی VirtualHost یا فایل ssl.conf اضافه شوند:

```
SSLCACertificateFile /etc/pki/tls/certs/ca-bundle.crt
SSLVerifyClient require
SSLVerifyDepth 10
```

بیاید این دستورات را به صورت دقیق بررسی کنیم:

• SSLCACertificateFile

این گزینه فایل حاوی گواهی‌های CA را معرفی می‌کند. سرور با استفاده از این گواهی‌ها، اعتبار گواهی ارائه‌شده توسط کلاینت را بررسی می‌کند. این فایل باید در قالب **PEM – Privacy Enhanced Mail** باشد و معمولاً شامل گواهی ریشه (Root CA) و گواهی‌های میانی (Intermediate CAs) است.

• SSLVerifyClient

این گزینه مشخص می‌کند که سرور چگونه باید با گواهی کلاینت برخورد کند. مقادیر قابل قبول برای این گزینه عبارت‌اند از:

- **none**: هیچ بررسی روی گواهی کلاینت انجام نمی‌شود.
- **optional**: اگر کلاینت گواهی ارائه دهد، بررسی می‌شود. در غیر این صورت هم اجازه دسترسی داده می‌شود.
- **require**: کلاینت حتماً باید یک گواهی معتبر ارائه دهد. در غیر این صورت اتصال رد خواهد شد.
- **optional_no_ca**: کلاینت باید گواهی ارائه کند، اما اگر گواهی توسط CA معتبر شناخته نشود، همچنان امکان ادامه اتصال وجود دارد (برای سناریوهای خاص آزمایشی استفاده می‌شود).

در اغلب سازمان‌ها و محیط‌های حساس، مقدار **require** انتخاب می‌شود تا دسترسی تنها به کلاینت‌های دارای گواهی معتبر محدود شود.

• SSLVerifyDepth

این گزینه تعیین می‌کند که سرور چه تعداد از زنجیره‌ی گواهی‌ها را باید بررسی کند. به عبارت دیگر، اگر کلاینت گواهی خود را از یک CA میانی دریافت کرده باشد، سرور باید تا گواهی ریشه را بررسی کند. مقدار 10 به معنی بررسی تا ۱۰ سطح از زنجیره است، اما برای گواهی‌های محلی معمولاً مقدار 1 یا 2 کافی است.

فصل ۳

331.3 سیستم فایل‌های رمزگذاری شده (Encrypted File Systems)

وزن: ۳ (یعنی اهمیت و ضریب این بخش در آزمون، ۳ است)

داوطلب باید توانایی راه‌اندازی و پیکربندی سیستم‌فایل‌های رمزگذاری شده را داشته باشد.

حوزه‌های اصلی دانش:

- درک مفاهیم رمزگذاری دستگاه‌های بلوکی (Block Device) و سیستم‌فایل‌ها
- استفاده از **dm-crypt** همراه با **LUKS1** برای رمزگذاری دستگاه‌های بلوکی
- استفاده از **eCryptfs** برای رمزگذاری سیستم‌فایل‌ها، از جمله دایرکتوری‌های خانگی و یکپارچه‌سازی با **PAM**
- آشنایی با **plain dm-crypt** (رمزگذاری ساده بدون متادیتا)
- آشنایی با قابلیت‌های **LUKS2**
- درک مفهومی **Clevis** برای دستگاه‌های **LUKS** و مکانیزم‌های **Clevis PIN** با استفاده از **TPM2** و **NBDE/Tang**

فهرست جزئی از ابزارها و فایل‌های مورد استفاده:

- ابزار **cryptsetup** و زیر دستورات مرتبط
- ابزار **cryptmount**
- فایل پیکربندی **etc/crypttab/**
- سرویس **ecryptfsd**
- دستورات خانواده **ecryptfs-***
- ابزارهای **mount.ecryptfs** و **umount.ecryptfs**
- ماژول **PAM** با نام **pam_ecryptfs**

مفاهیم رمزگذاری دیسک (Disk Encryption Concepts)

رمزگذاری داده‌ها در حالت ذخیره‌شده یا همان **Data at Rest** یکی از اساسی‌ترین نیازهای امنیتی در هر سازمان و شرکت مدرن محسوب می‌شود. در دنیای امروز که داده‌ها ارزشمندترین دارایی سازمان‌ها به حساب می‌آیند، محافظت از آن‌ها نه تنها یک گزینه بلکه یک ضرورت است. بسیاری از سازمان‌ها، به‌ویژه شرکت‌های بزرگ فناوری و مالی، بخش عمده‌ای از بودجه امنیتی خود را به رمزگذاری اختصاص می‌دهند تا از افشای داده‌ها جلوگیری کنند. با این حال، هنوز هم برخی شرکت‌ها و حتی کاربران شخصی به سمت رمزگذاری کامل دیسک نمی‌روند. یکی از دلایل اصلی این موضوع، نگرانی درباره‌ی کاهش عملکرد سیستم است. آن‌ها تصور می‌کنند که رمزگذاری با سربار پردازشی خود سرعت رایانه یا سرور را کاهش می‌دهد و بهره‌وری کلی سیستم پایین می‌آید.

این نگرانی تا حدودی در گذشته درست بود. زمانی که پردازنده‌ها توانایی کافی برای پردازش سریع الگوریتم‌های رمزنگاری نداشتند، واقعاً اجرای رمزگذاری می‌توانست سیستم را کند کند. اما امروزه، با وجود سخت‌افزارهایی که به‌طور خاص برای پردازش رمزنگاری ساخته شده‌اند، مانند **Intel AES-NI** یا **ARM Cryptography Extensions**، تأثیر رمزگذاری بر عملکرد سیستم بسیار ناچیز است. در بسیاری از موارد، کاربر حتی تفاوتی احساس نمی‌کند. بنابراین دلیل قدیمی "کند شدن سیستم" دیگر توجیه مناسبی برای عدم استفاده از رمزگذاری نیست.

کاربران می‌شد. LUKS این مشکل را حل کرد و امروز تقریباً در همه‌ی توزیع‌های لینوکسی به عنوان استاندارد اصلی رمزگذاری دیسک به کار گرفته می‌شود.

کار با LUKS در عمل

برای درک بهتر نحوه‌ی استفاده از LUKS، بیاید یک سناریوی واقعی را بررسی کنیم. فرض کنید ما به یک سیستم لینوکسی یک دیسک جدید ۱۰ گیگابایتی اضافه کرده‌ایم. ابتدا با استفاده از ابزارهایی مثل **fdisk** یا **parted** یک پارتیشن روی این دیسک ساخته‌ایم. (اگر آشنایی با این مرحله ندارید، می‌توانید به راهنمای **lpic1** مراجعه کنید که در آن نحوه‌ی ایجاد پارتیشن توضیح داده شده است.) حالا هدف ما این است که این پارتیشن جدید را با استفاده از LUKS رمزگذاری کنیم. ابزار اصلی برای این کار دستور **cryptsetup** است. در بسیاری از توزیع‌های لینوکسی مثل RHEL، CentOS یا RockyLinux، این ابزار به صورت پیش فرض نصب شده است. در غیر این صورت می‌توان آن را به سادگی نصب کرد.

مرحله ۱: ایجاد فرمت LUKS روی پارتیشن

اولین قدم این است که پارتیشن انتخابی (مثلاً /dev/sdb1) را با دستور زیر فرمت کنیم:

```
cryptsetup luksFormat /dev/sdb1
```

بعد از اجرای این دستور، یک هشدار مهم ظاهر می‌شود:

WARNING!

=====

This will overwrite data on /dev/sdb1 irrevocably.

این هشدار به ما می‌گوید که اجرای این دستور باعث از بین رفتن همه‌ی داده‌های موجود روی این پارتیشن خواهد شد. بنابراین قبل از ادامه، باید مطمئن شویم که پارتیشن موردنظر داده‌ی مهمی ندارد. سپس از ما خواسته می‌شود که برای این پارتیشن یک رمز عبور (Passphrase) انتخاب کنیم. این رمز عبور همان کلید اصلی است که برای باز کردن دیسک رمزگذاری شده استفاده خواهد شد.

مرحله ۲: باز کردن پارتیشن رمزگذاری شده

بعد از ایجاد فرمت LUKS، نوبت به باز کردن (unlock) پارتیشن می‌رسد. این کار با زیردستور **luksOpen** انجام می‌شود:

```
cryptsetup luksOpen /dev/sdb1 myencvol
```

در اینجا از ما رمز عبوری که قبلاً تنظیم کرده‌ایم خواسته می‌شود. بعد از وارد کردن صحیح رمز، پارتیشن باز شده و یک دیوایس مجازی جدید در مسیر **/dev/mapper/** ایجاد می‌گردد. در این مثال، نامی که انتخاب کرده‌ایم **myencvol** است. اگر دستور زیر را اجرا کنیم:

```
ls -l /dev/mapper/
```

می‌بینیم که یک لینک جدید به نام **myencvol** ایجاد شده است. این در واقع همان پارتیشن رمزگذاری شده است که حالا برای سیستم قابل استفاده می‌باشد.

مرحله ۳: ایجاد فایل سیستم روی پارتیشن رمزگذاری شده

وقتی پارتیشن باز شد، درست مانند یک دیسک عادی می‌توان روی آن فایل سیستم ایجاد کرد. برای مثال:

فصل ۵

332.1 سخت‌سازی میزبان (Host Hardening)

موضوع ۳۳۲: امنیت میزبان (Host Security)

وزن: ۵

شرح:

داوطلب باید بتواند کامپیوترهای لینوکسی را در برابر تهدیدات رایج ایمن‌سازی کند.

مباحث کلیدی:

- پیکربندی امنیت BIOS و بوت‌لودر (GRUB 2)
- غیرفعال کردن نرم‌افزارها و سرویس‌های بلااستفاده
- درک و حذف قابلیت‌های غیرضروری برای واحدهای systemd و کل سیستم
- درک و پیکربندی ASLR، DEP و Exec-Shield
- سیاه‌لیست و سفیدلیست کردن دستگاه‌های USB متصل با استفاده از USBGuard
- ایجاد یک CA برای SSH، صدور گواهی‌نامه‌های SSH برای کلیدهای میزبان و کاربر با استفاده از CA و پیکربندی OpenSSH جهت استفاده از این گواهی‌نامه‌ها
- کار با محیط‌های chroot
- استفاده از systemd units برای محدود کردن system call ها و قابلیت‌های یک پردازش
- استفاده از systemd units برای اجرای پردازش‌ها با دسترسی محدود یا بدون دسترسی به فایل‌ها و دستگاه‌های خاص
- استفاده از systemd units برای اجرای پردازش‌ها با دایرکتوری‌های /tmp و /dev اختصاصی و بدون دسترسی شبکه
- درک تأثیرات آسیب‌پذیری‌های Meltdown و Spectre در لینوکس و فعال/غیرفعال کردن روش‌های کاهش آن‌ها
- آگاهی از اگاهی از polkit
- آگاهی از مزایای امنیتی مجازی‌سازی و کانتینرسازی

امنیت کرنل (Kernel Security)

امنیت کرنل یکی از اساسی‌ترین موضوعات در بحث ایمن‌سازی سیستم‌عامل لینوکس است. کرنل در حقیقت قلب سیستم‌عامل محسوب می‌شود؛ یعنی همه چیز، از ساده‌ترین برنامه‌ها گرفته تا پیچیده‌ترین سرویس‌ها، برای دسترسی به منابع سخت‌افزاری باید با کرنل تعامل کنند. بنابراین اگر کرنل یا سرویس‌هایی که به آن وابسته هستند به درستی مدیریت و ایمن‌سازی نشوند، می‌توانند به یک نقطه ضعف بزرگ در کل ساختار امنیتی سیستم تبدیل شوند. به همین دلیل مفهومی به نام **سخت‌سازی کرنل (Kernel Hardening)** به وجود آمده است که شامل مجموعه اقداماتی می‌شود که با هدف کاهش سطح حمله، جلوگیری از سوءاستفاده‌ها و پایدارتر کردن سیستم انجام می‌گیرد.

در این بحث دو محور اساسی وجود دارد: نخست غیرفعال کردن نرم‌افزارها و سرویس‌های غیرضروری، و دوم محدود کردن استفاده از منابع سیستم توسط کاربران یا برنامه‌ها. هر یک از این دو مورد در ظاهر ساده به نظر می‌رسند، اما در عمل می‌توانند جلوی بسیاری از حملات و مشکلات جدی را بگیرند.

غیرفعال کردن نرم افزارها و سرویس های غیر ضروری

یکی از ابتدایی ترین گام های امنیتی در هر سیستم لینوکسی این است که بررسی کنیم چه سرویس ها و نرم افزارهایی در حال اجرا هستند و از میان آن ها کدام واقعاً برای عملکرد سرور لازم اند. هر سرویسی که بدون نیاز روشن مانده باشد، در حقیقت یک دروازه باز برای مهاجمان است. حتی اگر آن سرویس به ظاهر بی خطر به نظر برسد، در آینده ممکن است آسیب پذیری های امنیتی در آن کشف شود که هکرها بتوانند از آن سوءاستفاده کنند.

به عنوان **مثال**، در بسیاری از توزیع های لینوکس سرویس هایی مثل CUPS (مدیریت چاپگرها) یا Avahi-daemon (کشف خودکار دستگاه ها در شبکه محلی) به صورت پیش فرض فعال هستند. حالا تصور کنید این سرویس ها روی یک سرور وب در دیتاستر یا فضای ابری فعال بمانند. در چنین محیطی که هیچ چاپگری وجود ندارد و نیازی به کشف دستگاه ها در شبکه هم نیست، روشن ماندن این سرویس ها فقط به معنای افزایش سطح حمله است. یک مهاجم می تواند پورت باز مربوط به CUPS (یعنی ۶۳۱) را پیدا کند و اگر آسیب پذیری شناخته شده ای وجود داشته باشد، آن را هدف قرار دهد. به همین دلیل بهترین کار این است که این سرویس ها از همان ابتدا **متوقف و غیرفعال** شوند.

یکی از نکات مهمی که در اینجا باید یاد بگیریم این است که امنیت همیشه با کمینه سازی به دست می آید. هرچه سرویس های کمتری روی یک سیستم اجرا شوند، سطح حمله کاهش پیدا می کند. اصطلاحاً به این رویکرد «حداقل گرایی در امنیت» گفته می شود. یعنی فقط آنچه واقعاً نیاز داریم باید فعال باشد و باقی موارد باید حذف یا متوقف شوند. این قاعده ساده در عمل جلوی بسیاری از حملات را می گیرد.

برای مدیریت سرویس ها در لینوکس ابزارهای مختلفی وجود دارد. در توزیع های جدیدتر که از systemd استفاده می کنند، دستور `systemctl` کاربرد دارد. با آن می توانیم وضعیت یک سرویس را ببینیم، آن را متوقف کنیم و حتی برای همیشه غیرفعالش کنیم. به طور **مثال** اگر بخواهیم سرویس cups را غیرفعال کنیم کافی است دستورات زیر را اجرا کنیم:

```
systemctl stop cups
systemctl disable cups
```

اما در توزیع های قدیمی تر لینوکس که از SysVinit استفاده می کردند، ابزارهای متفاوتی مانند chkconfig و service به کار گرفته می شدند. برای **مثال**:

```
chkconfig cups off
service cups stop
```

در هر دو حالت هدف یکی است: سرویس های بلااستفاده نباید روی سیستم فعال باشند. اگر بخواهیم یک **مثال** واقعی از دنیای امنیت بیاوریم، در گذشته بارها گزارش شده که سرورهایی به دلیل روشن بودن سرویس های غیر ضروری مثل Avahi یا CUPS مورد نفوذ قرار گرفته اند. مهاجمان با یک اسکن ساده پورت ها متوجه باز بودن این سرویس ها شدند و از آسیب پذیری های شناخته شده آن ها بهره برداری کردند. مدیرانی که این سرویس ها را از ابتدا غیرفعال کرده بودند، هرگز درگیر چنین مشکلی نشدند. این نمونه ها نشان می دهند که گاهی اوقات ساده ترین اقدامات امنیتی مثل خاموش کردن یک سرویس می تواند از یک فاجعه بزرگ جلوگیری کند.

محدود کردن استفاده از منابع سیستم

دومین بخش از امنیت کرنل، مسئله محدود کردن منابع است. این موضوع شاید در نگاه اول به مدیریت عملکرد سیستم (Performance Management) مربوط به نظر برسد، اما در واقع یک لایه بسیار مهم از امنیت محسوب می شود.

تصور کنید روی سرور شما چندین کاربر اجازه اجرای برنامه دارند. اگر یکی از این کاربران - چه به عمد و چه سهواً - برنامه ای اجرا کند که منابع زیادی مصرف کند، ممکن است کل سیستم از کار بیفتد. به عنوان **مثال**، اگر او یک برنامه باگ دار بنویسد که حافظه بی پایان مصرف کند، سرور

مقدار +: غیرفعال (Disabled)

این مقدار ASLR را به طور کامل غیرفعال می‌کند. در این حالت:

- تمام برنامه‌ها در آدرس‌های ثابت و قابل پیش‌بینی بارگذاری می‌شوند.
- سیستم در برابر بسیاری از حملات حافظه آسیب‌پذیر می‌شود.
- این حالت تنها برای اهداف عیب‌یابی و توسعه باید استفاده شود.

روش‌های مدیریت پارامتر ASLR

استفاده از دستور sysctl :

```
# مشاهده وضعیت فعلی
sysctl kernel.randomize_va_space

# تغییر موقت به حالت کامل
sudo sysctl -w kernel.randomize_va_space=2

# تغییر به حالت محافظه کارانه
sudo sysctl -w kernel.randomize_va_space=1

# غیرفعال کردن موقت
sudo sysctl -w kernel.randomize_va_space=0
```

پیکربندی دائمی از طریق `/etc/sysctl.conf` :

برای اعمال تغییرات دائمی، باید فایل `/etc/sysctl.conf` را ویرایش کرد:

```
# اضافه کردن این خط برای فعال‌سازی دائمی ASLR
echo "kernel.randomize_va_space = 2" | sudo tee -a /etc/sysctl.conf

# اعمال تغییرات
sudo sysctl -p
```

روش جایگزین با استفاده از فایل‌های اختصاصی:

```
# ایجاد فایل اختصاصی در دایرکتوری sysctl
echo "kernel.randomize_va_space = 2" | sudo tee /etc/sysctl.d/99-aslr.conf
```

بهینه‌سازی ASLR برای محیط‌های مختلف

محیط‌های سرور پرترافیک:

در سرورهای تحت بار سنگین، ASLR ممکن است تاثیر جزئی بر عملکرد داشته باشد. در این موارد:

- استفاده از مقدار ۲ (حالت کامل) توصیه می‌شود

Chkrootkit و Rkhunter متخصصان کشف روتکیت

(Rootkit Hunter) Rkhunter و Chkrootkit دو ابزار تخصصی برای کشف روتکیت‌ها در سیستم‌های لینوکس هستند. روتکیت‌ها برنامه‌های مخربی هستند که برای پنهان کردن فعالیت‌های مخرب از دید مدیران سیستم طراحی شده‌اند.

- **Rkhunter** بر بررسی یکپارچگی فایل‌های باینری سیستم، بررسی فایل‌های configuration مشکوک و اسکن برای یافتن rootkit های شناخته شده تمرکز دارد.
- **Chkrootkit** با اجرای اسکریپت‌های تشخیصی، به جستجوی الگوهای رفتاری روتکیت‌ها و فایل‌های مخرب می‌پردازد.

بررسی عمیق AIDE

- معماری و مکانیزم عملکرد AIDE

AIDE از یک معماری ساده اما مؤثر برای تشخیص تغییرات غیرمجاز استفاده می‌کند. هسته مرکزی این ابزار شامل سه بخش اصلی است:

پایگاه داده امضاها:

این بخش شامل امضاهای دیجیتالی تمام فایل‌ها و دایرکتوری‌های تحت نظارت است. هر امضا با استفاده از الگوریتم‌های هش مختلف (MD5, SHA1, SHA256, etc.) محاسبه و ذخیره می‌شود.

موتور تحلیل و مقایسه:

این بخش مسئول مقایسه وضعیت فعلی سیستم با اطلاعات ذخیره شده در پایگاه داده است. این مقایسه می‌تواند بر اساس معیارهای مختلفی از جمله محتوا، metadata و attributes فایل‌ها انجام شود.

سیستم گزارش‌دهی:

این بخش تغییرات شناسایی شده را ثبت کرده و به مدیر سیستم گزارش می‌دهد.

امن سازی با AIDE و SELinux

یکی از جنبه‌های مهم AIDE، یکپارچه‌سازی عمیق آن با SELinux است. SELinux با اعمال کنترل دسترسی اجباری (Mandatory Access Control)، فرآیندهای AIDE را در برابر دستکاری و نفوذ محافظت می‌کند.

مکانیزم حفاظتی SELinux برای AIDE شامل:

- تعریف domain های امنیتی خاص برای فرآیندهای AIDE
- محدود کردن دسترسی این فرآیندها به منابع سیستم
- محافظت از پایگاه داده و فایل‌های پیکربندی AIDE
- جلوگیری از دستکاری در فرآیندهای نظارتی

سیاست امنیتی SELinux برای AIDE به اندازه‌ای انعطاف‌پذیر طراحی شده که به کاربران اجازه می‌دهد با حفظ امنیت، پیکربندی مورد نیاز خود را اعمال کنند.

نصب و راه‌اندازی اولیه AIDE

در برخی توزیع‌های لینوکس، AIDE به طور پیش‌فرض نصب نیست. برای نصب آن می‌توان از مدیریت بسته توزیع مربوطه استفاده کرد:

RHEL / CentOS / Rocky / AlmaLinux / Fedora

```
sudo dnf install aide
```

یا فرآیند مشکوکی را گزارش می‌دهد، یک مدیر سیستم باید با استفاده از روش‌های دیگر (مانند مقایسه با یک سیستم پایه سالم، بررسی لاگ‌ها و استفاده از ابزارهای مستقل) وجود تهدید را تأیید یا رد کند. همچنین شایان ذکر است که ابزارهای دیگر امنیتی مانند AIDE (یک سیستم تشخیص نفوذ مبتنی (on Integrity Checking) نیز می‌توانند با شناسایی تغییرات غیرمجاز در فایل‌های سیستمی حیاتی، به طور غیرمستقیم وجود روتکیت‌ها را آشکار سازند.

نصب و راه‌اندازی chkrootkit در توزیع‌های مختلف لینوکس

ابزار **chkrootkit** یکی از شناخته‌شده‌ترین اسکنرهای امنیتی در دنیای لینوکس است. همان‌طور که از نام آن پیداست، این ابزار برای **جستجوی روتکیت‌ها (Rootkits)** و آثار ناشی از حضور آن‌ها در سیستم طراحی شده است. روتکیت‌ها بدافزارهایی هستند که برای پنهان‌سازی فعالیت‌های غیرقانونی روی سیستم، فایل‌ها، پرتال‌ها و حتی سرویس‌های آلوده را از چشم مدیر سیستم مخفی می‌کنند. **chkrootkit** با روش‌های متنوع سعی می‌کند این نشانه‌ها را آشکار کند.

این پروژه یک ابزار **متن‌باز (Open Source)** است و توانایی شناسایی بیش از ۷۰ نوع **روتکیت شناخته‌شده** را دارد. لیست کامل این روتکیت‌ها در وب‌سایت رسمی پروژه قابل مشاهده است.

نصب در سیستم‌های مبتنی بر Ubuntu و Debian

در توزیع‌هایی مثل **Ubuntu** و **Debian**، نصب **chkrootkit** بسیار ساده است، چرا که این بسته در **مخازن اصلی (Main Repository)** موجود می‌باشد. کافی است از دستورات زیر استفاده کنید:

```
sudo apt update
sudo apt install chkrootkit
```

یا در نسخه‌های قدیمی‌تر:

```
sudo apt-get install chkrootkit
```

نصب در سیستم‌های مبتنی بر Red Hat، CentOS، Fedora و Rocky Linux

در توزیع‌هایی که بر پایه **Red Hat** ساخته شده‌اند، شرایط کمی متفاوت است. در بسیاری از موارد، بسته **chkrootkit** در **مخزن اصلی (Base Repository)** موجود نیست. بنابراین مدیر سیستم باید یکی از دو روش زیر را دنبال کند:

۱. استفاده از مخزن EPEL (Extra Packages for Enterprise Linux)

○ ابتدا مخزن EPEL را فعال کنید:

```
sudo dnf install epel-release
```

○ سپس **chkrootkit** را نصب کنید:

```
sudo dnf install chkrootkit
```

پس از نصب موفق، ساده‌ترین روش اجرای **chkrootkit** این است که آن را بدون هیچ آرگومانی فراخوانی کنید:

```
sudo chkrootkit
```

در این حالت، ابزار یک اسکن کامل روی سیستم انجام می‌دهد و نتیجه را در قالب یک خروجی نسبتاً مفصل (Verbose) نمایش می‌دهد. اما **chkrootkit** گزینه‌های خط فرمان متعددی دارد که به مدیر سیستم اجازه می‌دهد رفتار اسکن را کنترل کند.

گزینه‌های پرکاربرد chkrootkit

مدیریت منابع در لینوکس

در دنیای سیستم‌عامل لینوکس، مفهوم مدیریت منابع (System Resource Management) نقشی حیاتی در حفظ پایداری، امنیت و عملکرد بهینه سیستم ایفا می‌کند. برخلاف تصور رایج، یک سیستم عامل قدرتمند مانند لینوکس نه تنها باید بتواند منابع را به طور کارآمد تخصیص دهد، بلکه باید توانایی اعمال محدودیت‌های هوشمندانه بر مصرف این منابع را نیز داشته باشد. این محدودیت‌ها به ویژه در محیط‌های چندکاربره (Multi-user) و سرورهایی که میزبان سرویس‌های متعدد هستند، از اهمیت بالایی برخوردارند.

مثال عینی: تصور کنید یک کاربر روی سیستم با اجرای یک اسکریپت معیوب یا مخرب، شروع به ایجاد هزاران فرآیند کند یا فایل‌های عظیمی تولید نماید. بدون وجود مکانیزم‌های محدودکننده، این رفتار می‌تواند به سرعت تمام منابع سیستم (CPU)، حافظه، فضای دیسک (را مصرف کرده و باعث از کار افتادن سرویس‌های حیاتی یا حتی crash کل سیستم شود. ابزار ulimit در لینوکس دقیقاً برای جلوگیری از چنین سناریوهایی طراحی شده است.

دستور ulimit: کنترلگر منابع پوسته

دستور ulimit یک ابزار خط فرمان است که کنترل دقیقی بر منابع در دسترس برای پوسته (Shell) و فرآیندهای زاییده توسط آن ارائه می‌دهد. وقتی شما به عنوان یک کاربر در لینوکس وارد سیستم می‌شوید، پوسته شما مجموعه‌ای از محدودیت‌های پیش فرض را به ارث می‌برد که می‌توانند بسته به پیکربندی سیستم متفاوت باشند.

برای مشاهده وضعیت فعلی تمام محدودیت‌ها، از گزینه `-a` استفاده می‌کنیم:

```
[root@rocky-a ~]# ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority     (-e) 0
file size               (blocks, -f) unlimited
pending signals         (-i) 15465
max locked memory       (kbytes, -l) 64
max memory size         (kbytes, -m) unlimited
open files              (-n) 1024
pipe size               (512 bytes, -p) 8
POSIX message queues    (bytes, -q) 819200
real-time priority      (-r) 0
stack size              (kbytes, -s) 8192
cpu time                (seconds, -t) unlimited
max user processes      (-u) 15465
virtual memory          (kbytes, -v) unlimited
file locks              (-x) unlimited
```

برای کاهش خروجی `systemd-cgls` و مشاهده بخش مشخصی از سلسله مراتب، دستور زیر را اجرا کنید:

```
systemd-cgls name
```

که در آن `name` را با نام کنترلر منبعی که می‌خواهید بررسی کنید جایگزین می‌کنید.

ب: `systemd-cgtop` (نظارت بلادرنگ بر مصرف منابع)

دستور `systemd-cgls`

یک تصویر ثابت از سلسله مراتب `cgroup` ارائه می‌دهد. برای مشاهده یک گزارش پویا از `cgroups` در حال اجرا که بر اساس استفاده از منابع (CPU)، حافظه و I/O مرتب شده‌اند، از دستور زیر استفاده کنید:

```
systemd-cgtop
```

رفتار، آمارهای ارائه شده و گزینه‌های کنترل `systemd-cgtop` شبیه به ابزار `top` است. این ابزار به مدیران سیستم امکان می‌دهد تا به صورت بلادرنگ مصرف منابع هر `cgroup` را مشاهده کرده و مشکلات احتمالی را سریعاً شناسایی کنند.

نمایش زنده مصرف CPU/Memory/IO توسط `cgroup` ها

```
systemd-cgtop
```

نمایش منابع مصرفی با به‌روزرسانی هر ۵ ثانیه

```
systemd-cgtop -d 5
```

نمایش منابع مصرفی به صورت واحدهای `systemd`

```
systemd-cgtop -u
```

نمایش فقط ۱۰ گروه پرمصرف

```
systemd-cgtop -n 10
```

نمایش کامل درخت `cgroup` ها

```
systemd-cgls
```

نمایش فقط بخش `user.slice`

```
systemd-cgls user.slice
```

نمایش پردازه‌های یک کاربر خاص (UID 1000)

```
systemd-cgls user-1000.slice
```

نمایش پردازه‌های مربوط به سرویس `sshd`

```
systemd-cgls system.slice/sshd.service
```

نمایش پردازه‌های مربوط به سرویس `nginx`

فرض کنید شما روی یک سرور لینوکس کار می‌کنید که چندین کاربر مختلف دارد. اگر مکانیزم DAC وجود نداشت، هر کاربر می‌توانست آزادانه به فایل‌های دیگران دسترسی داشته باشد و حتی آن‌ها را تغییر دهد یا حذف کند. این موضوع نه تنها امنیت، بلکه پایداری کل سیستم را به خطر می‌اندازد.

DAC به مدیران سیستم این امکان را می‌دهد که:

- از حریم خصوصی کاربران محافظت کنند.
- منابع اشتراکی را به صورت امن مدیریت کنند.
- از بروز خطاهای ناخواسته (مانند حذف فایل‌های حیاتی توسط دیگر کاربران) جلوگیری کنند.

مدیریت دسترسی با دستور chmod

در سیستم‌عامل لینوکس، پرکاربردترین دستور برای تغییر مجوزهای دسترسی به فایل‌ها و پوشه‌ها، دستور **chmod** است. نام این دستور مخفف عبارت **Change Mode** به معنای "تغییر حالت" است. منظور از "حالت"، همان **سطح دسترسی (Permissions)** است که مشخص می‌کند چه کسی می‌تواند روی فایل یا پوشه چه **عملیاتی** انجام دهد. از آنجا که فایل‌ها و دایرکتوری‌ها پایه‌ی اصلی در لینوکس هستند، تسلط بر دستور chmod برای هر کاربر و مدیر سیستم یک ضرورت محسوب می‌شود. در این فصل، ما به طور کامل شیوه‌ی کار با chmod را توضیح می‌دهیم، تفاوت دو روش اصلی تغییر دسترسی‌ها (نمادین و اکتالی) را بررسی می‌کنیم، و **مثال‌ها**ی کاربردی برای هر روش ارائه می‌کنیم.

ساختار کلی دستور chmod

فرمت کلی دستور chmod به شکل زیر است:

```
chmod [reference][operator][mode] file...
```

- **reference مرجع:** (مشخص می‌کند تغییر دسترسی برای چه کسی اعمال شود).
- **operator عملگر:** (نوع عملیاتی که باید روی دسترسی انجام شود).
- **mode حالت:** (مجوزی که باید افزوده، حذف یا جایگزین شود).

تغییر دسترسی‌ها با کدهای نمادین (Symbolic Mode)

در این روش، به جای استفاده از اعداد، از حروف و نمادها استفاده می‌کنیم.

۱. دسته‌های کاربر (References)

- **u (user):** مالک فایل
- **g (group):** گروه فایل
- **o (others):** سایر کاربران
- **a (all):** همه (معادل ugo)

۲. عملگرها (Operators)

- متادیتاهای کاربری (مانند تگ گذاری فایل ها)
- اطلاعات سیستم فایل (مانند کش یا تنظیمات خاص)

استفاده می شوند.

دستورات مدیریت xattr

۱. مشاهده ویژگی های توسعه یافته

برای دیدن ویژگی های یک فایل:

```
getfattr -d filename
```

مثال:

```
getfattr -d test.txt
# output
# file: test.txt
user.comment="This is a test file"
```

۲. افزودن ویژگی جدید

```
setfattr -n user.comment -v "This is a test file" test.txt
```

- n- نام ویژگی (user.comment)
- v- مقدار ویژگی ("This is a test file")

۳. حذف ویژگی

```
setfattr -x user.comment test.txt
```

۴. مشاهده همه ویژگی ها

با گزینه -m می توان تمام ویژگی ها را نمایش داد:

```
getfattr -d -m - test.txt
```

انواع namespace در xattr

ویژگی های توسعه یافته در لینوکس در namespace های مختلف قرار می گیرند:

۱. user.*

- برای کاربران عادی قابل دسترسی است.
- معمولاً برای متادیتای سفارشی استفاده می شود.
- مثال: user.comment="Important file"

نمایش ACL یک فایل و فیلتر فقط دسترسی گروه

```
getfacl report.doc | awk '/^group:/'
```

نمایش ACL یک گروه خاص روی چند فایل

```
getfacl file1 file2 file3 | grep group:developers
```

نمایش سطح دسترسی گروه روی یک فایل اجرایی

```
getfacl /usr/bin/python3 | grep group
```

مقایسه ACL گروه‌ها بین دو فایل

```
diff <(getfacl file1 | grep group) <(getfacl file2 | grep group)
```

بخش ۴ – تنظیم ACL برای گروه‌ها با setfacl
دادن دسترسی خواندن به گروه developers

```
setfacl -m g:developers:r /var/www/app.conf
```

دادن دسترسی خواندن و نوشتن به گروه qa

```
setfacl -m g:qa:rw /home/test-results.txt
```

دادن دسترسی اجرا به گروه admins

```
setfacl -m g:admins:x /usr/local/bin/deploy.sh
```

دادن دسترسی کامل (rwx) به گروه sysadmin

```
setfacl -m g:sysadmin:rwx /etc/secure.conf
```

حذف دسترسی یک گروه خاص

```
setfacl -x g:qa /home/test-results.txt
```

دادن دسترسی همزمان به چند گروه

```
setfacl -m g:devops:rw,g:auditors:r /var/data/db.sqlite
```

دادن دسترسی پیش فرض برای گروه‌ها روی دایرکتوری

```
setfacl -d -m g:developers:rwx /srv/projects
```

دادن دسترسی نوشتن به گروه HR روی فایل حقوق

```
setfacl -m g:hr:w /home/hr/payroll.xlsx
```

دادن دسترسی اجرا به گروه guest روی یک اسکریپت

```
setfacl -m g:guest:x /usr/local/bin/guest-task.sh
```

۳. حالت‌های اجرایی (Modes): تعیین نحوه‌ی واکنش SELinux در مواجهه با نقض سیاست‌های امنیتی.

سیاست‌ها در SELinux

تعریف سیاست امنیتی

در SELinux، مفهوم سیاست امنیتی یا Policy اساس معماری امنیتی را تشکیل می‌دهد. سیاست‌ها مجموعه‌ای از قواعد هستند که مشخص می‌سازند کدام فرایندها مجاز به دسترسی به کدام منابع هستند و این دسترسی در چه سطحی انجام می‌شود. بدون وجود این سیاست‌ها، SELinux قادر به تصمیم‌گیری در خصوص مجاز یا غیرمجاز بودن یک عمل نخواهد بود.

انواع سیاست‌ها

سیاست‌های SELinux به چند دسته تقسیم می‌شوند که هر یک برای شرایط خاصی مناسب هستند:

- **سیاست هدفمند (Targeted Policy):**

در این حالت، تنها سرویس‌ها و فرایندهای حیاتی سیستم مانند Apache، SSH یا Bind تحت کنترل SELinux قرار می‌گیرند. سایر فرایندها صرفاً با مجوزهای سنتی لینوکس مدیریت می‌شوند. این نوع سیاست، به دلیل توازن میان امنیت و سادگی مدیریت، به‌طور پیش‌فرض در اکثر توزیع‌های لینوکس فعال است.

- **سیاست سخت‌گیرانه (Strict Policy):**

در این سیاست، تمام فرایندها و منابع سیستم مشمول کنترل SELinux هستند. امنیت فراهم‌شده در این حالت بسیار بالاست، اما مدیریت آن پیچیده‌تر بوده و ممکن است در برخی سرویس‌ها ایجاد اختلال کند.

- **سیاست چندسطحی (MLS – Multi-Level Security):**

این نوع سیاست برای محیط‌هایی با حساسیت بسیار بالا (مانند نهادهای نظامی و دولتی) طراحی شده است. در این حالت، هر فایل و فرایند بر اساس سطح محرمانگی (مانند محرمانه، سری و فوق‌سری) برچسب‌گذاری می‌شود و دسترسی‌ها مطابق این سطوح کنترل می‌گردد.

مثال

به‌عنوان نمونه، در سیاست‌های مربوط به سرویس وب‌سرور Apache تعریف می‌شود که این سرویس تنها مجاز به دسترسی به فایل‌هایی با برچسب `httpd_sys_content_t` است و اجازه‌ی دسترسی به فایل‌هایی با برچسب‌هایی مانند `ssh_home_t` را ندارد. این بدان معناست که حتی کاربر ریشه (Root) نیز نمی‌تواند این محدودیت را دور بزند، زیرا سیاست SELinux به‌طور اجباری اجرا خواهد شد.

برچسب‌گذاری (Labeling)

اهمیت برچسب‌گذاری

تمامی اشیای سیستم در SELinux دارای یک **برچسب امنیتی (Security Label)** هستند. این برچسب‌ها تعیین‌کننده‌ی هویت امنیتی هر شیء بوده و مبنای اصلی تصمیم‌گیری SELinux در مورد مجاز یا غیرمجاز بودن یک عملیات محسوب می‌شوند.

ساختار برچسب‌ها

یک برچسب امنیتی معمولاً به شکل زیر نمایش داده می‌شود:

```
user:role:type:level
```

- **User:** کاربر امنیتی تعریف‌شده در SELinux لزومی ندارد همان کاربر لینوکس باشد

- **Role:** نقش امنیتی مانند `sysadm_r` برای مدیر سیستم

۱,۳۳۴ امن سازی شبکه (Network Hardening)

موضوع: ۳۳۴ امنیت شبکه

وزن: ۴

شرح: داوطلبان باید توانایی ایمن سازی شبکه ها در برابر تهدیدات رایج را داشته باشند. این شامل تجزیه و تحلیل ترافیک شبکه برای گره ها و پروتکل های خاص می شود.

حوزه های کلیدی دانش:

- درک مکانیزم های امنیتی شبکه های بی سیم
- پیکربندی FreeRADIUS برای احراز هویت گره های شبکه
- استفاده از Wireshark و tcpdump برای تجزیه و تحلیل ترافیک شبکه، شامل فیلترها و آمار
- استفاده از Kismet برای تجزیه و تحلیل شبکه های بی سیم و ضبط ترافیک شبکه بی سیم
- شناسایی و برخورد با اعلان های مسیر یاب جعلی (rogue) و پیام های DHCP
- آگاهی از aircrack-ng و bettercap

فهرست جزئی از فایل ها، اصطلاحات و ابزارهای مورد استفاده:

- radiusd
- radmin
- radtest
- radclient
- radlast
- radwho
- radiusd.conf
- /etc/raddb/*
- wireshark
- tshark
- tcpdump
- kismet
- ndpmon

ابزار radlast

ابزار **radlast** مشابه دستور لینوکسی **last** عمل می‌کند، اما به جای خواندن اطلاعات از فایل‌های لاگ سیستم‌عامل، داده‌ها را از فایل‌های حسابداری **FreeRADIUS (Accounting)** دریافت می‌نماید.

کاربرد

- نمایش تاریخچه ورود و خروج کاربران
- بررسی مدت زمان اتصال هر کاربر
- کمک به شناسایی الگوهای استفاده یا موارد مشکوک

اجرای دستور زیر:

```
radlast
```

خروجی نمونه:

```
alice pts/0 192.168.1.10 Mon Oct 2 08:23 still logged in
bob pts/1 192.168.1.11 Sun Oct 1 17:05 - Sun Oct 1 18:15 (01:10)
charlie pts/2 192.168.1.12 Sun Oct 1 14:00 - Sun Oct 1 14:45 (00:45)
```

توضیحات:

- کاربر **alice** همچنان متصل است.
- کاربر **bob** یک ساعت و ده دقیقه متصل بوده است.
- کاربر **charlie** اتصال ۴۵ دقیقه‌ای داشته است.

ابزار radwho

ابزار **radwho** مشابه دستور لینوکسی **who** عمل می‌کند و کاربران در حال اتصال به سرور **RADIUS** را نمایش می‌دهد. این ابزار برای مانیتورینگ لحظه‌ای نشست‌ها (**Sessions**) بسیار کاربردی است.

کاربرد

- مشاهده کاربران فعال در لحظه
- بررسی نوع سرویس و زمان اتصال
- مانیتورینگ بار لحظه‌ای روی سرور

مثال

اجرای دستور زیر:

```
radwho
```

خروجی نمونه:

```
Login Name What TTY When From
alice - PPP ttyS0 08:23 192.168.1.10
bob - PPP ttyS1 17:05 192.168.1.11
```

توضیحات:

- ستون **Login** نام کاربر را نشان می‌دهد.
- ستون **From** نشانی IP کاربر یا دستگاهی است که اتصال از آن برقرار شده است.
- ستون **When** زمان آغاز نشست را مشخص می‌کند.

Zabbix

در سال‌های اخیر به‌طور گسترده در سازمان‌های بزرگ و متوسط به‌کار گرفته شده است. زیرا ترکیبی از ویژگی‌های قدرتمند سه ابزار قبلی را در خود دارد. برخلاف Nagios که بیشتر بر پایش سرویس‌ها متمرکز است یا Cacti که نمودارهای تاریخیچه ارائه می‌دهد، Zabbix تلاش می‌کند همه این قابلیت‌ها را یکجا و در یک پلتفرم یکپارچه در اختیار مدیر شبکه قرار دهد.

Zabbix یک نرم‌افزار متن‌باز و رایگان برای مانیتورینگ شبکه، سرورها، اپلیکیشن‌ها، سرویس‌ها و حتی ماشین‌های مجازی و Cloud است.

- Zabbix داده‌ها را به‌صورت لحظه‌ای (Real-time) جمع‌آوری می‌کند.
 - اطلاعات را در پایگاه داده‌های قدرتمند ذخیره می‌کند.
 - امکان ترسیم نمودارهای گرافیکی زیبا و ایجاد داشبوردهای تعاملی را دارد.
 - علاوه بر پایش وضعیت، از مکانیزم‌های هشداردهی پیشرفته و حتی اقدامات خودکار (Auto-remediation) پشتیبانی می‌کند.
- به عبارت دیگر، Zabbix تنها یک ابزار پایش ساده نیست، بلکه یک سیستم مدیریت هوشمند زیرساخت IT است.

ویژگی‌های کلیدی Zabbix

Zabbix مجموعه‌ای از قابلیت‌ها دارد که آن را از سایر ابزارهای سنتی متمایز می‌سازد:

۱. مانیتورینگ جامع (End-to-End Monitoring):
از شبکه و تجهیزات سخت‌افزاری گرفته تا اپلیکیشن‌ها و سرویس‌های Cloud، همه در Zabbix قابل پایش هستند.
۲. داشبوردهای تعاملی:
رابط کاربری Zabbix بسیار مدرن‌تر و کاربرپسندتر از ابزارهایی مثل Nagios یا Cacti است.
مدیر شبکه می‌تواند نمودارها، گراف‌ها، و ویجت‌های سفارشی بسازد.
۳. هشداردهی هوشمند:
Zabbix نه‌تنها می‌تواند در صورت بروز خطا ایمیل یا SMS ارسال کند، بلکه امکان اتصال به سیستم‌های پیام‌رسان مانند Slack، Microsoft Teams و Telegram را نیز دارد.
۴. قابلیت اقدامات خودکار (Auto-Remediation):
مثلاً اگر سرویس وب متوقف شود، Zabbix می‌تواند به‌طور خودکار دستور ری‌استارت سرویس را اجرا کند.
۵. مقیاس‌پذیری بالا (Scalability):
Zabbix می‌تواند هم برای یک شرکت کوچک با چند سرور استفاده شود و هم برای یک سازمان جهانی با هزاران سرور و دستگاه.
۶. ذخیره‌سازی قدرتمند داده‌ها:
برخلاف ابزارهایی مثل Nagios که داده‌ها را ساده ذخیره می‌کنند، Zabbix از پایگاه داده‌های قدرتمند مانند MySQL، PostgreSQL یا Oracle برای ذخیره و تحلیل اطلاعات استفاده می‌کند.

سه جدول اصلی در iptables وجود دارد:

۱. **Filter Table** جدول پیش فرض برای فیلتر کردن بسته‌ها.
۲. **NAT Table** جدول مخصوص ترجمه‌ی آدرس شبکه (برای پیاده‌سازی SNAT و DNAT).
۳. **Mangle Table** برای پردازش‌های خاص روی بسته‌ها (مانند تغییر برخی فیلدها).

زنجیره‌ها در iptables

زنجیره‌ها نقاطی هستند که در آن‌ها می‌توان قوانین را تعریف کرد. مهم‌ترین زنجیره‌ها عبارتند از:

- **prerouting**: قبل از تصمیم‌گیری درباره‌ی مسیریابی بسته‌ها.
- **input**: بسته‌هایی که مقصد آن‌ها خود سیستم است.
- **forward**: بسته‌هایی که از سیستم عبور می‌کنند اما مقصدشان جای دیگری است.
- **output**: بسته‌هایی که از خود سیستم خارج می‌شوند.
- **postrouting**: درست قبل از خروج بسته از سیستم.

این زنجیره‌ها بسته به نوع جدول (Filter, NAT, Mangle) کاربردهای مختلفی دارند. برای مثال، ترجمه‌ی آدرس در زنجیره‌ی POSTROUTING جدول NAT انجام می‌شود.

اهداف (Targets) و سیاست‌ها (Policies) در iptables

وقتی یک بسته وارد زنجیره‌ی iptables می‌شود، هر قانون (Rule) بررسی می‌شود. اگر بسته با شرایط یک قانون مطابقت داشته باشد، آن قانون روی بسته اعمال می‌شود. نتیجه‌ی اعمال قانون، همان چیزی است که به آن **Target** یا «هدف» گفته می‌شود.

اهداف اصلی در iptables

- **ACCEPT**: اجازه‌ی عبور بسته داده می‌شود.
- **DROP**: بسته بدون هیچ پاسخی حذف می‌شود. (کاربر سمت مقابل هیچ اطلاعی از رد شدن بسته نخواهد داشت)
- **REJECT**: بسته رد می‌شود اما به فرستنده پیام خطا بازگردانده می‌شود (مثلاً ICMP Port Unreachable).
- **LOG**: اطلاعات بسته در فایل لاگ ذخیره می‌شود، ولی پردازش آن ادامه پیدا می‌کند.
- **RETURN**: پردازش به زنجیره‌ی قبلی باز می‌گردد.

سیاست پیش فرض (Default Policy)

هر زنجیره در iptables می‌تواند یک سیاست پیش فرض داشته باشد. اگر بسته با هیچ قانونی در آن زنجیره مطابقت نداشته باشد، سیاست پیش فرض اعمال می‌شود.

نمونه:

```
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT ACCEPT
```

در این مثال، سیاست پیش فرض برای بسته‌های ورودی و عبوری «DROP» است، اما برای بسته‌های خروجی «ACCEPT» است. این روش یکی از متداول‌ترین رویکردهای امنیتی است:

بسته به توزیع لینوکس، ابزارهای مختلفی برای ذخیره‌ی قوانین وجود دارد:

- در دبیان/اوبونتو:

```
iptables-save > /etc/iptables/rules.v4
ip6tables-save > /etc/iptables/rules.v6
```

- در ردهت/سنت‌اواس:

```
service iptables save
```

بازیابی قوانین

برای بازیابی قوانین:

```
iptables-restore < /etc/iptables/rules.v4
```

این روش باعث می‌شود بعد از هر بار بوت، سیستم به طور خودکار قوانین قبلی را بارگذاری کند.

فیلترینگ پیشرفته و امنیت بیشتر

در بسیاری از سناریوها تنها ACCEPT یا DROP کافی نیست. iptables قابلیت‌های پیشرفته‌ای دارد که امکان اعمال محدودیت‌های دقیق‌تر را فراهم می‌کند.

نمونه‌هایی از ماژول‌های پرکاربرد

۱. **Limit**: محدود کردن تعداد بسته‌ها در یک بازه‌ی زمانی.

مثال: حداکثر یک بسته ICMP در ثانیه:

```
iptables -A INPUT -p icmp -m limit --limit 1/s -j ACCEPT
```

۲. **state / conntrack**: بررسی وضعیت اتصال.

مثال: اجازه دادن فقط به بسته‌های مرتبط (Established, Related):

```
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

۳. **mac**: فیلتر کردن بر اساس آدرس MAC.

```
iptables -A INPUT -m mac --mac-source 00:11:22:33:44:55 -j ACCEPT
```

۴. **Recent**: جلوگیری از حملات Brute-force.

مثال: محدود کردن تعداد تلاش برای ورود به SSH:

```
iptables -A INPUT -p tcp --dport 22 -m recent --set --name SSH
iptables -A INPUT -p tcp --dport 22 -m recent --update --seconds 60 --hitcount 5 --name SSH -j DROP
```

فیلترینگ IPv6 با ip6tables

همان‌طور که iptables برای IPv4 استفاده می‌شود، ابزار **ip6tables** برای مدیریت بسته‌های IPv6 طراحی شده است.

کاربرد ip6tables مشابه iptables است و تقریباً همان دستورات را دارد، تنها تفاوت در پروتکل مقصد است.

مثال: مسدود کردن همه‌ی ورودی‌ها به پورت ۲۲ در IPv6:

```
ip6tables -A INPUT -p tcp --dport 22 -j DROP
```

حالت‌های کاری IPSec

IPSec می‌تواند در دو حالت اصلی عمل کند:

۱. Transport Mode

در این حالت، فقط بخش Payload بسته IP رمزگذاری یا احراز هویت می‌شود، اما هدر اصلی IP دست‌نخورده باقی می‌ماند. این حالت بیشتر برای ارتباط مستقیم بین دو میزبان استفاده می‌شود.

۲. Tunnel Mode

در حالت تونل، کل بسته IP درون یک بسته‌ی جدید قرار می‌گیرد (Encapsulation) و سپس رمزگذاری می‌شود. این روش در ارتباطات بین دو شبکه (Site-to-Site VPN) کاربرد دارد، زیرا کل بسته‌ها از دید اینترنت عمومی مخفی می‌مانند.

کاربردهای IPSec

- اتصال ایمن میان دفاتر سازمان (Site-to-Site VPN)
- دسترسی کاربران راه‌دور به شبکه‌ی سازمان (Remote Access VPN)
- رمزگذاری ارتباطات میان سرورها و دیتاسترها
- استفاده در پروتکل‌های دیگر مانند IPv6 که IPSec بخشی از استاندارد آن محسوب می‌شود.

نصب و راه‌اندازی IPSec در لینوکس

برای پیاده‌سازی IPSec در لینوکس معمولاً از بسته‌هایی مانند **Libreswan** یا **strongSwan** استفاده می‌شود. این بسته‌ها ابزارهای کاربرپسندی برای مدیریت تنظیمات و ارتباطات IPSec فراهم می‌کنند.

برای نمونه، روی Rocky Linux می‌توان Libreswan را نصب کرد:

```
dnf install libreswan -y
```

فایل‌های پیکربندی IPSec معمولاً در مسیر `/etc/ipsec.conf` قرار دارند. یک نمونه‌ی ساده از فایل پیکربندی می‌تواند به شکل زیر باشد:

```
config setup
    protostack=netkey
    plutodebug=all
```

```
conn myvpn
    left=192.168.1.1
    leftsubnet=192.168.1.0/24
    right=203.0.113.1
    rightsubnet=10.0.0.0/24
    auto=start
    authby=secret
    keyexchange=ike
    ike=aes256-sha2;modp1024
```

حملات (DDoS) (Distributed Denial of Service)

در این حمله، مهاجم تعداد زیادی درخواست جعلی را از طریق هزاران دستگاه آلوده (Botnet) به یک سرور ارسال می‌کند. این حجم عظیم از ترافیک باعث می‌شود منابع سرور مصرف شود و وبسایت یا سرویس هدف از دسترس خارج شود. تفاوت اصلی بین DoS و DDoS در مقیاس آن‌هاست. حمله DoS از یک منبع واحد انجام می‌شود، در حالی که حمله DDoS از منابع متعدد در سراسر جهان انجام می‌گیرد.

حملات مرد میانی (Man-in-the-Middle)

در این حمله، مهاجم میان ارتباط دو سیستم قرار می‌گیرد و داده‌های در حال انتقال را شنود یا تغییر می‌دهد. برای مثال، هنگام اتصال کاربر به یک وبسایت، مهاجم می‌تواند ترافیک بین مرورگر و وبسایت را رهگیری کرده و داده‌های حساس مانند رمز عبور یا شماره کارت بانکی را سرقت کند.

ارتقای دسترسی (Privilege Escalation)

ارتقای دسترسی زمانی رخ می‌دهد که یک کاربر یا نرم‌افزار بتواند به سطحی از دسترسی دست پیدا کند که در حالت عادی نباید به آن دسترسی داشته باشد. این ارتقا می‌تواند به دو شکل باشد:

- **عمودی: (Vertical)** کاربر معمولی به سطح مدیر ارتقا می‌یابد.
- **افقی: (Horizontal)** کاربر به داده‌ها یا حساب کاربری دیگر کاربران دسترسی پیدا می‌کند.

تزریق (SQL Injection)

SQL زبان استاندارد مدیریت پایگاه داده است. در حمله SQL Injection، مهاجم کد مخرب SQL خود را به ورودی‌های وبسایت تزریق می‌کند. اگر وبسایت به درستی ایمن‌سازی نشده باشد، این کدها مستقیماً در پایگاه داده اجرا می‌شوند و مهاجم می‌تواند اطلاعات حساس کاربران را سرقت یا تغییر دهد.

DHCP جعلی (Rogue DHCP)

DHCP وظیفه تخصیص خودکار آدرس IP و اطلاعات شبکه به دستگاه‌ها را بر عهده دارد. یک DHCP جعلی می‌تواند توسط کاربر یا بدافزار در شبکه راه‌اندازی شود. اگر دستگاه‌ها به این سرور جعلی متصل شوند، آدرس‌های اشتباه دریافت کرده و در نتیجه ترافیک آن‌ها می‌تواند توسط مهاجم شنود یا دستکاری شود.

نقاط دسترسی جعلی (Rogue Access Point)

یک نقطه دسترسی جعلی زمانی ایجاد می‌شود که فردی یک Access Point غیرمجاز در شبکه راه‌اندازی کند. این نقطه می‌تواند باعث شود کاربران به شبکه مهاجم متصل شوند و تمام داده‌هایشان به سرقت برود.

جعل ARP و NDP

در شبکه‌های IPv4، پروتکل ARP وظیفه تطبیق IP با MAC Address را بر عهده دارد. در IPv6 نیز پروتکل NDP همین نقش را دارد. هر دو این پروتکل‌ها در برابر جعل آسیب‌پذیر هستند. مهاجم می‌تواند با ارسال پاسخ‌های جعلی، جریان ترافیک شبکه را به سمت خود هدایت کرده و نقش مرد میانی را ایفا کند.

contact me:

HosseinSeilani

Designer, Developer and Linux sysadmin

seilany.ir

learninghive.ir

emperor-os.ir

Hubuntu.ir

predator-os.ir

telegram: @seilany



Founder and Developer of Emperor-OS, Hubuntu and Predator-OS. I bring significant experience as a Linux/Windows sysadmin and graphical web design, UX/UI to the Open Source Community.



Emperor-OS



Predator-OS