

امنیت در سیستم‌عامل‌های

# یونیکس و لینوکس

مهندس مجید داوری دولت‌آبادی

عضو گروه امنیت GrayHat Hackers

Security Information Assets

انتشارات پندار پارس

سرشناسه	: داوری دولت‌آبادی، مجید، ۱۳۵۹ -
عنوان و نام پدیدآور	: امنیت در سیستم‌عامل‌های یونیکس و لینوکس / مجید داوری دولت‌آبادی.
مشخصات نشر	: تهران : پندار پارس، ۱۳۹۳.
مشخصات ظاهری	: ۵۵۲ ص. : مصور، جدول.
شابک	: ۲۵۰۰۰۰ ریال ۹۷۸-۶۰۰-۶۵۲۹-۵۶-۱
وضعیت فهرست نویسی	: فیبا
یادداشت	: کتابنامه.
موضوع	: سیستم عامل لینوکس
موضوع	: سیستم عامل یونیکس
موضوع	: سیستم‌های عامل (کامپیوتر)
موضوع	: کامپیوترها -- ایمنی اطلاعات
رده بندی کنگره	: ۷۶/۷۶۵۸ ۱۳۹۳ ۹۴۵۲/س
رده بندی دیویی	: ۴۳۳/۰۰۵
شماره کتابشناسی ملی	: ۲۴۵۰۲۴۵

#### انتشارات پندار پارس



دفتر فروش: انقلاب، ابتدای کارگر جنوبی، کوی رشتچی، شماره 14، واحد 16 [www.pendarepars.com](http://www.pendarepars.com)  
 تلفن: 66572335 - تلفکس: 66926578 همراه: 09122452348 [info@pendarepars.com](mailto:info@pendarepars.com)

نام کتاب	: امنیت در سیستم‌عامل‌های یونیکس و لینوکس
ناشر	: انتشارات پندار پارس
ترجمه و تالیف	: مهندس مجید داوری دولت‌آبادی
چاپ دوم	: اردیبهشت 93
شمارگان	: 500 نسخه
لیتوگرافی	: ترام‌سنج
چاپ، صحافی	: فرشویه، خیام
قیمت	: 35000 تومان
شابک	: 978-600-6529-56-1

\* هرگونه کپی برداری، تکثیر و چاپ کاغذی یا الکترونیکی از این کتاب بدون اجازه ناشر تخلف بوده و پیگرد قانونی دارد\*

## فهرست

3	فصل نخست: مروری بر امنیت در سیستم‌های عامل کدباز.....
4	1-1 مکانیزم‌های مختلف در بهبود امنیت لینوکس.....
6	1-2 بهره‌گیری از خطاها برای امن‌سازی لینوکس.....
8	1-3 توصیه‌های مهم امنیتی در مورد یونیکس و لینوکس.....
9	1-4 مفاهیم پایه‌ای در امنیت سیستم‌های عامل کدباز (لینوکس و یونیکس).....
10	1-4-1 مفهوم TCPWrapper.....
11	1-4-2 مفهوم دایمون xinetd.....
13	1-4-3 مفهوم دایمون identd.....
14	1-5 مقاوم‌سازی سرورهای لینوکسی.....
22	1-6 مهم‌ترین نقاط آسیب‌پذیر یونیکس و لینوکس.....
23	1-7 امنیت سیستم‌عامل یونیکس.....
23	1-7-1 امنیت یونیکس.....
23	1-7-2 سیستم حفاظت یونیکس.....
25	1-7-3 تحلیل امنیتی یونیکس.....
27	1-7-4 آسیب‌پذیری‌های یونیکس.....
29	1-8 تفاوت‌های اصلی دو سیستم‌عامل یونیکس و لینوکس.....
۳۰	1-9 مراجع این فصل.....
31	فصل دوم: امن‌سازی ساختارهای مدیریتی و سیستمی.....
31	2-1 بررسی درستی بسته‌های نرم‌افزاری.....
31	2-2 پیکربندی TCP Wrapperها.....
33	2-3 ابزار Bastille.....
33	2-4 انواع هسته‌ها از نظر نوع عملکرد در لینوکس.....
33	2-4-1 هسته‌های نوع پیش‌فرض و سفارشی.....
34	2-4-2 هسته‌های نوع پایدار و درحال توسعه.....
34	2-5 ماژول‌ها در هسته‌ی لینوکس.....
36	2-6 غیرفعال کردن ماژول‌های هسته غیرضروری.....
36	2-7 آسیب‌پذیری ارتقای سطح دسترسی به‌روز Page Fault در هسته‌ی لینوکس.....
37	2-8 امنیت هسته.....
38	2-9 ابزارهای امن‌سازی هسته.....
40	2-10 مفهوم LSM.....
41	2-11 ماژول‌های امنیتی لینوکس.....
41	2-11-1 ساختار امنیتی Flask.....
44	2-11-2 اعمال تایپ یا TE (DTE).....
45	2-11-3 کنترل دسترسی بر مبنای نقش یا RBAC.....
46	2-11-4 ماژول امنیتی SELinux.....
52	2-11-5 ماژول امنیتی Capability.....
54	2-12 استفاده از ماکروها.....
54	2-13 معرفی ماژول‌های دیگر.....
55	2-13-1 Openwall ماژول امنیتی.....
55	2-13-2 DTE ماژول امنیتی.....
55	2-13-3 LIDS ماژول امنیتی.....
56	2-14 مفهوم رویه‌ی از راه دور (RPC).....
58	2-15 ضدویروس در لینوکس.....

58	.....AVAST	2-15-1 ضدویروس
59	.....BitDefender	2-15-2 ضدویروس
59	.....ClamAV	2-15-3 ضدویروس
59	.....AVG	2-15-4 ضدویروس
60	.....sysctl	2-16 دستور
62	.....	2-17 لیست کنترلی امن سازی سیستم عامل لینوکس
66	.....	2-18 مراجع این فصل
67	<b>فصل سوم: احراز هویت و ساختارهای کنترل دسترسی</b>	
67	.....	3-1 تعیین سطوح دسترسی و مجوزهای کاربران در لینوکس
69	.....	3-2 اجرای دستورها با مجوز دیگر کاربران در لینوکس
71	.....	3-3 محدودسازی دسترسی در یونیکس و لینوکس
72	.....	3-4 دستورهای متدنخست برای تنظیم و تغییر مجوزها در لینوکس
75	.....	3-5 حسابهای کاربری معمولی و مدیر سیستم
76	.....	3-6 ماژولهای تصدیق اصالت در لینوکس
83	.....shadow و passwd	3-7 ساختار فایل های
84	.....	3-8 دستورهای مدیریتی برای مدیریت کاربران
88	.....	3-9 حسابهای کاربری با رمز عبور ضعیف
92	.....	3-10 تغییر گذرواژه های حساب کاربری ریشه در لینوکس
93	.....	3-11 تأمین گذرواژه های قوی
96	.....	3-12 سیستم حسابرسی در لینوکس
102	.....	3-13 مراجع این فصل
103	<b>فصل چهارم: امن سازی سرویس های وب</b>	
103	.....	4-1 امنیت در سرورهای وب
104	.....	4-2 فرض های امنیتی سرور وب
105	.....Apache	4-3 انتخاب سیستم عامل پیش از نصب سرور وب
105	.....Apache	4-4 آماده سازی میزبان برای نصب امن سرور
105	.....Apache	4-4-1 امنیت میزبان
107	.....Apache	4-4-2 بارگذاری و اعتبارسنجی
108	.....SSL	4-4-3 ایجاد گواهی
110	.....Apache	4-5 نصب امن سرویس دهنده ی وب
110	.....	4-5-1 عملیات مربوط به نصب
110	.....	4-5-2 کد منبع یا نسخه ی اجرایی
111	.....	4-5-2-1 دریافت و نصب کد منبع
112	.....	4-5-2-2 دریافت وصله ها
113	.....	4-5-3 نسخه ی اجرایی ایستا یا ماژول های پویا
113	.....Apache	4-5-4 درهای پشتی
114	.....	4-5-4-1 مکان پوشه ها
115	.....Apache	4-5-5 نصب سرویس دهنده ی
115	.....	4-5-5-1 پیکربندی کد منبع
116	.....Apache	4-5-5-2 کامپایل و نصب
116	.....Apache	4-5-5-3 آزمایش درستی نصب
116	.....	4-5-5-4 انتخاب ماژول ها برای نصب
118	.....	4-5-6 تنظیم حساب کاربری سرویس دهنده ی وب
119	.....Apache	4-5-7 تنظیم مجوزهای فایل اجرایی
119	.....Apache	4-5-8 قراردادن در محبس
121	.....chroot	4-5-8-1 ابزارهای مورد نیاز

121	.....chroot	4-5-8-2	مثالی از کاربرد
122	.....ltd	4-5-8-3	استفاده از ld برای مشخص کردن وابستگی ها
123	.....strace	4-5-8-4	استفاده از strace برای مشاهدهی پردازشها
124	.....chroot	4-5-8-5	استفاده از chroot برای جای دادن Apache در محبس
124	.....	4-5-8-6	جای دادن فایل های کاربر، گروه و تفکیک کنندهی نام در محبس
125	.....Apache	4-5-8-7	اتمام انتقال Apache به محیط محبس
125	.....php	4-5-9	آماده کردن php برای کار در محبس
126	.....perl	4-5-10	آماده سازی perl برای کار در محیط محبس
127	.....	4-5-11	مراقبت از مشکلات کوچک محبس
127	.....chroot(2)	4-5-12	استفاده از وصله ی (2)chroot
128	.....mod_chroot یا mod_security	4-5-13	استفاده از ماژول mod_chroot یا mod_security
129	.....1.x	4-5-13-1	سرویس دهندهی Apache شاخه ی 1.x
129	.....2.x	4-5-13-2	سرویس دهندهی Apache شاخه ی 2.x
129	.....Apache	4-6	فایل های حیاتی در پیکربندی سرور وب Apache
130	.....Apache	4-7	کنترل سرویس دهندهی Apache
131	.....httpd.conf	4-8	فایل پیکربندی httpd.conf
134	.....Apache	4-9	پیکربندی امن سرویس دهندهی وب Apache
135	.....	4-9-1	پیش فرض های پیکربندی امن
135	.....	4-9-2	گزینه های رهنمود
136	.....AllowOverride	4-9-2-1	رهنمود AllowOverride
137	.....CGI	4-9-3	فعال سازی اسکریپت های CGI
138	.....	4-9-4	واقع نگاری
138	.....	4-9-5	پیکربندی تنظیمات محدودکنندهی سرویس دهنده
140	.....IP	4-9-5-1	محدودیت دسترسی به وسیله ی IP
140	.....	4-9-6	جلوگیری از فاش شدن اطلاعات
142	.....	4-9-7	تغییر هویت سرویس دهندهی وب
143	.....	4-9-8	تغییر فیلد سرآیند سرویس دهنده
143	.....	4-9-8-1	تغییر نام در کد منبع
144	.....mod_security	4-9-8-2	تغییر نام به وسیله ی mod_security
145	.....mod_headers	4-9-8-3	تغییر نام به وسیله ی mod_headers در Apache نسخه دو
145	.....	4-9-8-4	حذف محتویات پیش فرض
146	.....LDAP و SSL	4-10	پیکربندی امن سرویس دهندهی Apache با پشتیبانی SSL و LDAP
146	.....Apache	4-10-1	پیکربندی امن Apache
146	.....Apache	4-10-1-1	تغییر نسخه ی Apache
146	.....	4-10-1-2	پیکربندی نرم افزار Apache برای کامپایل
148	.....Apache	4-10-1-3	کامپایل سرور Apache
149	.....Apache	4-10-1-4	نصب سرور Apache
149	.....SSL	4-10-2	نصب گواهی SSL
149	.....Apache	4-10-2-1	تنظیم شاخه ی گواهی Apache
149	.....Apache	4-10-3	تنظیم سرور Apache
149	.....httpd.conf	4-10-3-1	پیکربندی httpd.conf
152	.....ssl.conf	4-10-3-2	پیکربندی ssl.conf
152	.....Apache	4-10-3-3	حذف فایل های پیش فرض Apache
152	.....	4-10-3-4	تنظیم مجوز فایل ها و شاخه های سرور
153	.....	4-10-4	تنظیمات نهایی و نصب سرور
153	.....Apache	4-10-4-1	تغییر اسکریپت راه اندازی Apache

153	4-10-4-2 بررسی پیکربندی سرور با راهاندازی
153	4-10-4-3 تنظیم LDAP
154	4-10-4-4 اجرای سرور Apache
154	4-11 کنترل دسترسی‌ها در سرویس‌دهنده‌ی Apache
156	4-11-1 کنترل دسترسی به سایت
156	4-11-1-1 استفاده از ماژول mod_access
157	4-11-1-2 استفاده از ماژول mod_rewrite
157	4-12 فایل پیکربندی modules.conf
158	4-13 ماژول‌های امنیتی سرویس‌دهنده‌ی Apache
158	4-13-1 تنظیمات سرور وب
158	4-13-2 ماژول mod_rewrite
160	4-13-3 ماژول mod_security
164	4-14 تنظیم امن سرویس‌دهنده‌ی وب Apache با رویکرد اصلاحی
170	4-15 برخی نقاط آسیب‌پذیر در سرویس‌دهنده‌ی وب Apache
171	4-15-1 سیستم‌عامل‌های در معرض تهدید
171	4-15-2 نحوه‌ی حفاظت در مقابل نقطه آسیب‌پذیر
173	4-16 حمله به تصدیق اصالت در سرویس‌دهنده‌ی Apache
174	4-16-1 فناوری‌های تصدیق اصالت
174	4-16-2 ضعف‌های طراحی در مکانیزم‌های تصدیق اصالت
175	4-16-2-1 گذرواژه‌های نامناسب
175	4-16-2-2 ورود به روش Brute-force
176	4-16-2-3 پیام‌های خطای طولانی
177	4-16-2-4 انتقال آسیب‌پذیر حساب‌های کاربری
178	4-16-2-5 مکانیزم تغییر گذرواژه
179	4-16-2-6 مکانیزم مربوط به فراموشی گذرواژه
179	4-16-2-7 مکانیزم یادآوری حساب کاربری
180	4-16-2-8 اعتبارسنجی ناقص حساب‌های کاربری
180	4-16-2-9 نام کاربری غیریکتا
181	4-16-2-10 نام‌های کاربری قابل پیش‌بینی
181	4-16-2-11 گذرواژه‌های آغازین قابل پیش‌بینی
181	4-16-2-12 توزیع نامن حساب‌های کاربری
182	4-16-3 ضعف‌های پیاده‌سازی تصدیق اصالت
182	4-16-3-1 مکانیزم ورود Fail-Open
183	4-16-3-2 شکست در مکانیزم ورود چندمرحله‌ای
185	4-16-3-3 ذخیره نامن حساب‌های کاربری
185	4-16-4 امن‌سازی تصدیق اصالت
185	4-16-4-1 استفاده از حساب‌های کاربری قوی
186	4-16-4-2 مدیریت امن حساب‌های کاربری
186	4-16-4-3 اعتبارسنجی صحیح حساب‌های کاربری
187	4-16-4-4 جلوگیری از افشای اطلاعات
188	4-16-4-5 جلوگیری از حمله‌های نوع ورود به‌زور (Brute-force)
189	4-16-4-6 جلوگیری از سوءاستفاده از مکانیزم تغییر گذرواژه
190	4-17 تنظیم ثبت وقایع در راستای امن‌سازی Apache
191	4-17-1 محول کردن ثبت وقایع به برنامه‌های خارجی
192	4-17-2 ملاحظات امنیتی در فایل‌های ثبت وقایع
192	4-17-2-1 ثبت وقایع و دسترسی ریشه (root)

193	4-17-2-2	ثبت وقایع در فایل‌های متنی قابل تغییر
193	4-17-2-3	مشکل امنیتی در برنامه‌های ثبت وقایع
193	4-17-2-4	ثبت وقایع و فضای دیسک
194	4-17-2-5	ثبت وقایع غیرقابل اعتماد
194	4-17-3	خواندن فایل‌های ثبت وقایع
194	4-17-3-1	The Error Log ثبت وقایع خطا
195	4-17-3-2	The Access Log ثبت وقایع دسترسی یا
196	4-18	مشاهده‌ی رویدادها در سرویس‌دهنده‌ی Apache
197	4-18-1	استفاده از دایمون syslogd
198	4-18-1-1	error_log استفاده از syslog برای رویدادنگاری
199	4-18-1-2	access_log استفاده از syslog برای رویدادنگاری
200	4-18-2	استفاده از syslog-ng
200	4-19	به‌کارگیری ماژول mod_security در امنیت Apache
204	4-20	تنظیم قوانین فیلتر ماژول mod_security در امنیت Apache
211	4-21	نصب و راه‌اندازی mod_Security نسخه دو بر روی Apache
214	4-22	مراجعه این فصل
215		<b>فصل پنجم: امن‌سازی سرویس پست‌الکترونیکی</b>
215	5-1	امنیت در پست‌الکترونیکی
215	5-1-1	پست‌الکترونیکی چگونه کار می‌کند
215	5-1-2	فقدان امنیت در پست‌الکترونیکی
216	5-1-3	تهدیدهای امنیتی در ارتباطات پست‌الکترونیکی
217	5-1-4	امن‌سازی پست‌الکترونیکی توسط SSL و TLS
218	5-1-5	حفاظت از حریم خصوصی توسط SMTP ناشناس
219	5-1-6	رمزگذاری نامتقارن و پست‌الکترونیکی (PGP و S/MIME)
219	5-1-7	معرفی PGP
220	5-1-8	مشکلات تعاملی
221	5-2	آسیب‌پذیری‌های موجود در بسته‌ی پست‌الکترونیکی Sendmail
223	5-3	بسته‌ی پست‌الکترونیکی Postfix
224	5-4	ایمن‌سازی سرور پست‌الکترونیکی
224	5-4-1	ایمن‌سازی سرورهای خانواده لینوکس
225	5-4-2	ایمن‌سازی بسته‌های پست‌الکترونیکی (Sendmail)
226	5-4-3	ایمن‌سازی بسته‌ی پست‌الکترونیکی (Qmail)
226	5-4-4	ایمن‌سازی بسته‌ی پست‌الکترونیکی (Postfix)
227	5-4-5	پرهیز از Open Relay
228	5-4-6	بلوکه کردن Spamها
229	5-4-7	فیلترکردن ویروس‌ها
230	5-5	اطلاعات پایه‌ی امنیتی در عامل انتقال نامه‌الکترونیکی Qmail
230	5-5-1	عامل انتقال نامه‌الکترونیکی Qmail
232	5-5-2	نکته‌هایی برای استفاده از Qmail
232	5-5-3	اشکالات گزارش شده در Qmail
233	5-6	پی‌کردنی عامل انتقال نامه‌الکترونیکی Qmail
233	5-6-1	پی‌کردنی Qmail
237	5-7	لیست کنترلی امن‌سازی عامل انتقال نامه‌الکترونیکی Qmail
238	5-8	مراجعه این فصل
239		<b>فصل ششم: امن‌سازی سرویس‌های فایل و به‌اشتراک‌گذاری آن</b>
239	6-1	اصول آغازین سطوح دسترسی فایل در یونیکس و لینوکس

239	6-1-1 مجوزهای دستیابی پرونده‌ها
241	6-1-2 نکاتی در مورد مجوز دسترسی به برخی فایل‌های کلیدی
241	6-1-3 ویژگی‌های پرونده‌ها
244	6-2 کنترل دسترسی به فایل‌ها و ویژگی‌های آنها
245	6-3 ابزار Tripwire
246	6-3-1 Tripwire چگونه کار می‌کند؟
247	6-4 سیستم‌های فایل رمزنگاری شده
248	6-5 امن‌سازی سرویس‌دهنده‌ی FTP
250	6-5-1 ابزارهای مرتبط برای آزمایش FTP
250	6-5-2 حمله‌های محتمل بر علیه FTP
251	6-5-3 معرفی vsftpd
252	6-5-4 نصب و استفاده از vsftpd
252	6-5-5 پیکربندی vsftpd
254	6-5-6 نکات امنیتی تکمیلی
254	6-6 لیست کنترلی امن‌سازی سرویس‌دهنده‌ی FTP
255	6-7 پروتکل SFTP و FTPS
255	6-7-1 Secure FTP یا (SFTP)
255	6-7-2 استفاده از SSL برای امن کردن سرویس‌دهنده‌های FTP
256	6-7-3 تفاوت SFTP و FTPS
256	6-8 پروتکل ساده انتقال فایل (TFTP)
257	6-9 سرویس‌دهنده‌ی اشتراک فایل (Samba)
257	6-9-1 فایل پیکربندی smb.conf
261	6-9-2 کار با سرویس‌دهنده‌ی Samba
262	6-10 سیستم‌فایل شبکه (NFS)
262	6-11 عدم پیکربندی مناسب سرویس‌های NIS/NFS
265	6-12 مراجع این فصل
267	<b>فصل هفتم: امن‌سازی پایگاه‌های داده‌ی رایج در یونیکس و لینوکس</b>
267	7-1 ضعف‌های امنیتی در پایگاه‌های داده
270	7-2 محافظت از پایگاه‌های داده
272	7-3 معماری پایگاه‌داده‌ی MySQL
273	7-4 امن‌سازی پایگاه‌داده‌ی MySQL
274	7-4-1 نصب سرور MySQL به‌فرم امن
276	7-4-2 پیکربندی سرور MySQL به‌فرمی امن
278	7-4-3 حذف اشیاء غیرضروری در پایگاه‌های داده‌ی MySQL
279	7-4-4 ایمن‌سازی حساب‌های کاربری
280	7-4-5 استفاده از احراز هویت ایمن برای تصدیق هویت کاربران
282	7-4-6 کنترل دسترسی کاربران
284	7-4-7 ذخیره‌سازی امن اطلاعات
284	7-4-8 پشتیبان‌گیری منظم از اطلاعات
287	7-4-9 فعال‌سازی گزارش‌های ثبت وقایع
289	7-5 ملاحظات امنیتی در پایگاه‌داده‌ی MySQL
293	7-6 پیاده‌سازی مکانیزم‌های امنیتی در MySQL
296	7-7 بهره‌گیری از ابزارهای امنیتی برای MySQL
298	7-8 دیواره‌آتش پایگاه‌داده‌ی MySQL
299	7-9 مکانیزم‌های امنیتی در PostgreSQL
303	7-10 کنترل دسترسی از طریق فایل‌های مخصوص پیکربندی



305	7-11 مکانیزم SSL در PostgreSQL
306	7-12 برقراری ارتباط محلی امن در PostgreSQL
308	7-13 امن‌سازی پایگاه‌های داده PostgreSQL
313	7-14 امنیت در پایگاه‌های داده‌ی Oracle
316	7-15 کنترل دسترسی در پایگاه‌داده‌ی Oracle
320	7-16 محافظت از پایگاه‌های داده Oracle
321	7-17 ایمن‌سازی ساختاری در پایگاه‌داده‌ی Oracle
325	7-18 بهره‌گیری از مکانیزم‌های امنیتی در پایگاه‌داده‌ی Oracle
330	7-19 امن‌سازی پایگاه‌داده‌ی Berkeley DB
332	7-20 مراجع این فصل
333	<b>فصل هشتم: اصول امنیت در DNS</b>
334	8-1 آسیب‌پذیری‌های موجود در BIND DNS
336	8-2 آسیب‌پذیری عدم سرویس‌دهی در BIND
336	8-3 امنیت در سرویس DNS
338	8-4 لیست کنترلی امن‌سازی سرویس نام‌دانه (DNS)
340	8-5 مراجع این فصل
341	<b>فصل نهم: امن‌سازی دیگر سرویس‌ها و پروتکل‌های رایج</b>
341	9-1 پروتکل آسان مدیریت شبکه (SNMP) و امن‌سازی آن
345	9-2 مدیریت سیستم گزارش‌گیری و ثبت وقایع (syslog)
346	9-2-1 تفاوت Syslog با SNMP در عملیات کنترل و نظارت
346	9-2-2 اجزای یک سرور Syslog پیشرفته
347	9-2-3 آشنایی با ساختار پیغام‌های Syslog
350	9-2-4 فایل‌های مهم مخصوص ثبت وقایع (Log)
353	9-2-5 بهره‌گیری از دستورهای faillog و logger
353	9-3 پیکربندی و مدیریت تحلیل Log
353	9-3-1 تحلیل‌گر Syslog
354	9-3-1-1 پیکربندی syslog برای ورودهای محلی و راه‌دور
355	9-3-2 بسته‌های باز مفید برای بازرسی فایل‌های ثبت وقایع (Log)
355	9-3-3 گردش Log
356	9-3-4 تحلیل‌گر Log
357	9-4 بررسی Log‌های ورودهای نهایی
358	9-5 امن‌سازی در پروتکل DHCP
358	9-5-1 حمله‌های نوع DHCP Starvation
358	9-5-2 حمله‌های نوع DHCP Spoofing
359	9-6 شناسایی DHCP در لینوکس
360	9-7 امن‌سازی پروتکل NTP
360	9-7-1 جعل و دست‌کاری بسته‌های NTP
360	9-7-2 حمله‌ی سرریز بافر
361	9-7-3 حمل تکرار
361	9-8 پیشنهاد برای امن‌سازی سیستم‌عامل لینوکس
364	9-9 شناسایی آسیب‌پذیری‌ها و حفره‌های امنیتی در لینوکس
367	9-10 مدیریت آسیب‌پذیری‌ها و وصله‌ها در لینوکس
367	9-11 مراجع این فصل
369	<b>فصل دهم: بهره‌گیری از دیوارهای آتش</b>
369	10-1 سرویس‌های دیوار آتش
370	10-2 انواع دیوار آتش

370	.....	10-3 ابزار ITables
371	.....	10-4 مفاهیم پایه در ITables
373	.....	10-5 عبارات و اصطلاحات ITables
375	.....	10-6 ساختار ITables
376	.....	10-6-1 حرکت در میان جدول‌ها و زنجیره‌ها
380	.....	10-6-2 مفهوم ماشین‌وضعیت و ورودی‌های Contrack
381	.....	10-6-3 وضعیت‌های مختلف یک بسته
382	.....	10-6-3-1 اتصال‌های نوع TCP
383	.....	10-6-3-2 اتصال‌های نوع UDP
383	.....	10-6-3-3 اتصال‌های نوع ICMP
384	.....	10-6-4 امکانات ابزار ITables
384	.....	10-6-4-1 دستور iptables-save
386	.....	10-6-4-2 دستور iptables-restore
386	.....	10-6-5 ایجاد قوانین در دیوارآتش
394	.....	10-6-6 تفاوت ITables و IPChains
394	.....	10-6-7 مسدود کردن یک آدرس IP
395	.....	10-7 راه‌اندازی و تنظیم دیواره‌های آتش در محیط گرافیکی
396	.....	10-7-1 بهره‌گیری از محیط گرافیکی دیوارآتش برای تنظیم
396	.....	10-7-1-1 عبارت فعال و غیرفعال کردن دیوارآتش
397	.....	10-7-1-2 گزینه‌ی سرویس‌های قابل اعتماد (Trusted Services)
397	.....	10-7-1-3 گزینه‌ی دیگر پورت‌ها (Other Ports)
397	.....	10-7-1-4 گزینه‌ی واسط‌های قابل اعتماد (Trusted Interfaces)
397	.....	10-7-1-5 گزینه‌ی masquerading
397	.....	10-7-1-6 گزینه‌ی Port forwarding
397	.....	10-7-1-7 گزینه‌ی ICMP Filter
398	.....	10-8 استفاده از ITables برای تنظیمات خاص
400	.....	10-9 جدول‌های موجود در ITables
401	.....	10-10 آزمون نفوذپذیری دیواره‌های آتش با ابزار FTTester
402	.....	10-10-1 نصب FTTester
402	.....	10-10-2 پیکربندی آغازین
403	.....	10-10-3 استفاده از FTTester در آزمون دیوارآتش
404	.....	10-10-3-1 گام نخست
404	.....	10-10-3-2 گام دوم
405	.....	10-10-3-3 گام سوم
407	.....	10-11 دیوارآتش IPFW در FreeBSD
407	.....	10-11-1 پیکربندی در هسته
407	.....	10-11-2 فایل /etc/rc.conf
409	.....	10-11-3 دستور ipfw
410	.....	10-11-4 نحوه‌ی نوشتن قوانین
410	.....	10-11-4-1 فیلد CMD
410	.....	10-11-4-2 فیلد RULE_NUMBER
410	.....	10-11-4-3 فیلد ACTION
410	.....	10-11-4-4 فیلد LOGGING
411	.....	10-11-4-5 فیلد SELECTION
412	.....	10-11-5 جدول‌ها
415	.....	10-12 دیوارآتش ipf

416	10-13 مراجع این فصل
417	فصل یازدهم: سیستم‌های تشخیص نفوذ، Honeypot و مانیتورینگ
417	11-1 سیستم‌های تشخیص نفوذ
418	11-1-1 سیستم‌های تشخیص نفوذ تحت شبکه (NIDS)
419	11-1-2 سیستم‌های تشخیص نفوذ تحت میزبان (HIDS)
420	11-1-3 طبقه‌بندی IDSها برپایه‌ی قابلیت‌ها
422	11-2 سیستم تشخیص نفوذ Snort
423	11-2-1 اجزای نرم‌افزار Snort
423	11-2-1-1 بخش گردآوری اطلاعات
423	11-2-1-2 بخش تجزیه و تحلیل کننده‌ی آغازین
423	11-2-1-3 بخش موتور شناسایی
424	11-2-1-4 بخش خروجی هشدار
424	11-3 نصب و پیکربندی Snort
425	11-3-1 نرم‌افزارهای پیش‌نیاز Snort
425	11-3-1-1 Libpcap
426	11-3-1-2 PCRE
426	11-3-1-3 Libnet
426	11-3-1-4 Barnyard
426	11-3-2 نصب Snort
426	11-3-2-1 نصب Snort با استفاده از بسته‌ی RPM
427	11-3-2-2 راه‌اندازی، متوقف کردن و راه‌اندازی دوباره‌ی Snort
427	11-3-2-3 نصب Snort با استفاده از کد منبع
427	11-3-2-4 کامپایل و نصب
429	11-3-3 پیکربندی Snort
432	11-3-4 واسط‌های کاربری Snort
437	11-4 معماری و امنیت در ابزار Snort
437	11-4-1 تأمین امنیت Snort
437	11-4-2 تشخیص سیستم Snort روی شبکه
438	11-4-3 حمله‌های تهدیدکننده‌ی Snort
439	11-4-4 معماری Snort در شبکه
440	11-5 سیستم‌های تشخیص نفوذ معروف
441	11-6 ابزارهای مفید و رایج برای کار با Snort
441	11-6-1 ابزار SnortSnarf
442	11-6-2 ابزار Snort Alert Monitor
443	11-6-3 ابزار Guardian (Active Response for Snort)
444	11-6-4 ابزار BASE
444	11-7 Honeypot چیست؟
445	11-8 انواع Honeypot
445	11-8-1 Honeypot‌های با تعامل کم
447	11-8-2 Honeypot‌های با تعامل متوسط
447	11-8-3 Honeypot‌های با تعامل زیاد
449	11-8-4 تقسیم‌بندی Honeypot‌ها از نظر کاربرد
449	11-8-4-1 Production Honeypot
450	11-8-4-2 Research Honeypot
450	11-9 مکانیزم‌های تحلیل اطلاعات در Honeypot‌ها
451	11-9-1 فایل‌های Log مخصوص دیوارآتش

451	سیستم تشخیص نفوذ (IDS)	11-9-2
451	فایل‌های Log سیستم	11-9-3
452	جرم‌شناسی سیستم قربانی	11-9-4
452	جرم‌شناسی پیشرفته سیستم قربانی	11-9-5
452	مکانیزم‌های گردآوری اطلاعات در Honeypot ها	11-10
453	مبتنی بر میزبان	11-10-1
453	مبتنی بر شبکه	11-10-2
453	مبتنی بر مسیر یاب / دروازه (Gateway)	11-10-3
454	گام‌های راه‌اندازی و به‌کارگیری یک Honeypot	11-11
454	انتخاب سخت‌افزار برای میزبان	11-11-1
454	نصب سیستم‌عامل	11-11-2
455	معماری شبکه	11-11-3
455	هشدارها و تشخیص نفوذ	11-11-4
455	ابزارهای کنترل و نظارت شبکه	11-12
459	مراجع این فصل	11-13
461	<b>فصل دوازدهم: سیستم‌های پراکسی</b>	
462	نصب سرویس‌دهنده‌ی Squid	12-1
463	برخی از پارامترهای مهم در Squid	12-2
464	پیکربندی سرویس‌دهنده‌ی Squid	12-3
465	تنظیم پورت سرویس‌دهنده‌ی Squid	12-3-1
465	تنظیم محل ذخیره‌سازی شیء‌های Cache شده	12-3-2
465	شناسه‌های گروه و کاربر	12-3-3
466	ارسال نامه‌الکترونیکی برای مدیریت Cache	12-3-4
466	اطلاعات ورود به سرویس FTP	12-3-5
466	تنظیمات کنترل دسترسی	12-3-6
468	ثبت وقایع در سرویس‌دهنده‌ی Squid	12-3-7
469	مفهوم upstream Proxy در سرویس‌دهنده‌ی Squid	12-3-8
469	به‌اشتراک‌گذاری سرورهای Cache با کمک Squid	12-3-9
469	پیاپی‌سازی سرور پراکسی به‌فرم شفاف	12-3-10
470	راه‌اندازی سرویس‌دهنده‌ی Squid	12-4
470	امن‌سازی سرویس‌دهنده‌ی Squid	12-5
472	سرویس‌دهنده‌ی Squid در FreeBSD	12-6
472	پیکربندی Squid	12-6-1
473	راه‌اندازی و متوقف کردن سرور Squid	12-6-2
473	تنظیم کلاینت‌ها	12-6-3
473	بررسی فایل پیکربندی	12-6-4
480	قابلیت لیست‌های کنترل دسترسی در Squid در FreeBSD	12-6-5
483	برچسب‌های cache_log و cache_store_log, cache_access_log	12-6-6
483	برچسب ftp_user	12-6-7
483	برچسب dns_nameservers	12-6-8
483	برچسب auth_param	12-6-9
484	پارامتر authenticate_ttl	12-6-10
484	پارامتر request_header_max_size	12-6-11
484	بررسی برچسب‌هایی با تعیین مدت زمان	12-6-12
484	برچسب‌های امنیتی در Squid	12-6-13
485	اجرای Squid در حالت Intercept یا Transparent	12-6-14

486	12-7 مراجع این فصل
487	فصل سیزدهم: کانال‌های امن و امن‌سازی ساختار آنها
487	13-1 OpenSSL و عملیات مربوط به گواهینامه‌ها
487	13-1-1 تولید درخواست امضای گواهینامه
488	13-1-2 تولید گواهینامه‌ی self-signed با در اختیار داشتن CSR
489	13-1-3 مشاهده‌ی محتوای CSR
489	13-1-4 مشاهده‌ی محتوای یک گواهینامه
489	13-1-5 مشاهده‌ی امضاکننده‌ی یک گواهینامه
490	13-1-6 تولید کلید خصوصی و رمزنگاری/رمزگشایی آن
490	13-1-7 تبدیل گواهینامه با فرمت PEM به گواهینامه با فرمت PKCS12
491	13-1-8 تبدیل گواهینامه با فرمت PKCS12 به گواهینامه با فرمت PEM
491	13-1-9 فعال کردن OpenSSL به‌عنوان سرویس‌دهنده
492	13-1-10 فعال کردن OpenSSL به‌عنوان سرویس‌گیرنده
492	13-2 حمله‌های بر علیه SSL و امن‌سازی آن در لینوکس
492	13-2-1 حمله‌ها علیه SSL
492	13-2-1-1 حمله‌ی Man in the Middle
493	13-2-1-2 حمله‌ی Brute-force بر روی کلید نشست
493	13-2-1-3 حمله‌ی از کاراندازی سرویس
493	13-2-1-4 حمله‌ی تحلیل ترافیک
494	13-2-1-5 آسیب‌پذیری Debian OpenSSL در مکانیزم تولید اعداد تصادفی
494	13-2-2 نحوه‌ی تشخیص آسیب‌پذیری‌های OpenSSL و حفاظت از آن
495	13-3 ساختار VPN در یونیکس و لینوکس
495	13-3-1 نصب PPTP
496	13-3-2 نصب L2TP
496	13-3-3 نصب OpenVPN
498	13-4 نصب امن سرویس‌دهنده‌ی SSH
498	13-4-1 انواع بسته
498	13-4-1-1 بسته‌های دودویی
499	13-4-1-2 بسته‌های کد منبع
499	13-4-2 پیش‌نیازها
499	13-4-3 دریافت بسته
499	13-4-4 نصب بسته‌ی دودویی
499	13-4-4-1 بررسی درستی بسته
500	13-4-4-2 نصب بسته
500	13-4-4-3 نصب بسته‌ی کد منبع
501	13-4-4-4 خارج کردن بسته از حالت فشرده
501	13-4-4-5 پیکربندی برای کامپایل
505	13-4-4-6 کامپایل
506	13-4-4-7 نصب بسته
506	13-4-4-8 جایگاه فایل‌ها
506	13-4-4-9 فایل‌های پیکربندی
506	13-4-4-10 فایل‌های اجرایی
507	13-5 امن‌سازی سرویس‌دهنده‌ی SSH در لینوکس
508	13-5-1 بهره‌گیری از SSH به‌عنوان یک سرویس امن
508	13-5-2 تولید کلیدهای RSA
509	13-5-3 بهره‌گیری از دستور ssh-agent

510	.....	SSH	امن سرویس دهنده‌ی	13-5-4
511	.....	SSH	تغییر دادن فایل شنود	13-5-4-1
511	.....	SSH	مجاز دانستن پروتکل شماره دو برای	13-5-4-2
511	.....	SSH	تنها مجاز دانستن کاربران خاص برای ورود به سیستم از طریق	13-5-4-3
512	.....	SSH	ایجاد یک نشان سفارشی	13-5-4-4
512	.....	DSA	تصدیق اصالت با استفاده از کلید عمومی	13-5-4-5
513	.....	SSH	برقراری یک نشست با سرویس راه دور	13-5-5
513	.....	sftp و scp	انتقال فایل با برنامه‌های	13-5-6
514	.....	SSH	راه اندازی سرویس دهنده‌ی	13-5-7
514	.....	SSH	نمایش یا عدم نمایش آدرس IP نهایی متصل شده به	13-5-8
515	.....	SSH	پیاده‌سازی تنظیمات امنیتی بیشتر در	13-5-9
516	.....	SSH	نحوه‌ی حفاظت درمقابل نقاط آسیب‌پذیر سرویس دهنده‌ی	13-5-10
517	.....		سرویس‌های متن ساده در لینوکس	13-6
519	.....	GPG	ساختار ابزار	13-7
520	.....	GPG	نصب	13-7-1
520	.....		ایجاد کلیدهای رمزنگاری	13-7-2
523	.....		کار با کلیدهای رمزنگاری تولید شده	13-7-3
526	.....		امضای کلیدهای رمزنگاری	13-7-4
529	.....		رمزنگاری و امضای یک فایل	13-7-5
530	.....	GnuPG	ابزارها برای کار با	13-7-6
530	.....		ابزارهای گرافیکی	13-7-6-1
530	.....		سرویس گیرنده‌های پست الکترونیکی	13-7-6-2
531	.....	GnuPG	ابزارهای دیگر برای کار با	13-7-6-3
531	.....	GPG	محل قرارگیری فایل‌های ابزار	13-7-7
532	.....		مراجع این فصل	13-8

## سخنی با خوانندگان

این روزها امنیت اطلاعات در هر شاخه‌ای از علم، دارای جایگاه ویژه‌ای می‌باشد که روز به روز بر ارزش آن افزوده می‌شود. هم اینک تقریباً تمامی صنایع و حرفه‌ها به نوعی نیازمند امنیت اطلاعات و ایجاد مکانیزم‌هایی جهت محرمانگی آنها هستند. امنیت در یک سیستم‌عامل نیز جزو موارد اساسی و پایه‌ای در یک شبکه‌ی کامپیوتری است که هر مدیر سیستمی باید به آن توجه ویژه داشته باشد. هم اکنون سیستم‌های عامل یونیکس و لینوکس از نظر امنیت نسبت به سیستم‌های عامل ویندوز در جایگاه بهتری قرار دارند، اما باز هم دارای نقاط ضعف و آسیب‌پذیری‌های خاص خود می‌باشند که امن‌سازی آنها می‌تواند یک سیستم‌عامل قابل قبول را از دیدگاه امنیتی پدید آورد؛ هرچند همان‌گونه که می‌دانید، امنیت یک موضوع نسبی است و به هیچ‌عنوان ساختاری مطلق به‌خود نخواهد گرفت، اما می‌توان تا حد امکان و قابل قبول و با کمک ساختارها و اصول امنیتی، بستر اصلی آن یعنی سیستم‌عامل را امن ساخت. همان‌گونه که گفته شد، امروزه وجود یک سیستم‌عامل امن و پایدار، از ارکان اصلی و پایه‌ای یک بستر و شبکه‌ی کامپیوتری محسوب می‌شود که هرگونه اختلال در آن حتی جزئی، ممکن است موجب تخریب و سرقت داده‌ها و یا از بین رفتن همه‌ی اطلاعات موجود بر روی آن سیستم گردد. از این‌رو استفاده از یک سیستم‌عامل پایدار، مطمئن و امن می‌تواند در بسترهای مختلف کامپیوتری به کاربران، مدیران و متخصصان فناوری اطلاعات کمک به‌سزایی نماید و آنها را در مسیر سرویس‌دهی و نگه‌داری از اطلاعات حساس سازمان‌ها یاری کند.

اینجانب به‌عنوان عضو کوچکی از خانواده‌ی بزرگ سیستم‌های عامل کدباز، درصدد گردآوری و تألیف کتابی مرجع به‌منظور آشنایی کامل برخی متخصصان، دانشجویان و مدیران شبکه با اصول امن‌سازی و امنیت سیستم‌های عامل مبتنی بر یونیکس و لینوکس بودم تا آنها را با اصول امنیت در سیستم‌های عامل کدباز و همچنین اهمیت امنیت در آنها آشنا و آگاه سازم. (گرچه مدیران و متخصصان، حکم اساتید اینجانب را دارند، اما به حکم وظیفه بر خود لازم دانستم که این آگاه‌سازی را انجام دهم). پیشاپیش تمام کاستی‌های آن را می‌پذیرم و ضمن پوزش از اساتید، متخصصان، دانشجویان و مدیران عزیز، انتقادات و راهنمایی‌های دلسوزانه‌ی آنها را به دیده‌ی منت پذیرا هستم.

( m\_Davari@TOP-co.ir )

( m\_Davary@Parshack.zzn.com )

**درباره‌ی این کتاب**

شیرازه‌ی اصلی این کتاب برگرفته از کتاب‌ها و منابع معتبر درباره‌ی امنیت و امن‌سازی سیستم‌های عامل یونیکس و لینوکس است که با تجربیات اینجانب آمیخته شده است، که دخل و تصرفی نیز با آن همراه بوده است. هدف از تألیف این کتاب آشنایی بیشتر کارشناسان و مدیران فناوری اطلاعات با آسیب‌پذیری‌های موجود در سیستم‌های عامل کدباز (یونیکس و لینوکس) است. همچنین راه‌کارهای مقابله با این نوع آسیب‌پذیری‌ها که موجب بروز حملات مختلف بر علیه این نوع سیستم‌های عامل می‌شوند، در قالب فصل‌های این کتاب بررسی و ارائه شده است. این کتاب اطلاعات مفیدی در اختیار راهبر سیستم‌های یونیکسی و لینوکسی می‌گذارد و وی با کمک این کتاب می‌تواند سیستم‌عامل خود را که معمولاً برای مواردی چون سرور و سامانه‌های امنیتی مورد استفاده قرار می‌گیرد، امن نماید تا در مقابل انواع حملات متداول و غیرمتداول ایمن گردد.

پس از سپاس و ستایش به درگاه پروردگار، از همه‌ی دوستان و اساتید عزیزی که مهربانانه دست مرا در انجام اینکار ناچیز فشردند، سپاسگزاری می‌کنم. بر خود لازم می‌دانم از زحمات بی‌دریغ استاد محترم سرکار خانم مهندس سیده پونه مرتضویان تشکر و قدردانی نمایم. زحمات خاضعانه‌ی ایشان سهم بزرگی در تهیه و تدوین این کتاب داشته است.

در پایان از جناب آقای مهندس حسین یعسوبی و تمامی همکارانشان که زحمت چاپ کتاب را متقبل شده‌اند، صمیمانه قدردانی می‌نمایم.

یارب چو به وحدتت یقین می‌دارم

ایمان به تو عالم آفرین می‌دارم

دارم لب خشک و دیده‌ی تر ببذیر

کز خشک و تر جهان همین می‌دارم

(مجید داوری دولت آبادی - زمستان 1392)



## فصل نخست

### مروری بر امنیت در سیستم‌های عامل کد باز

سیستم عامل، یکی از عناصر چهارگانه در یک سیستم کامپیوتری است که دارای نقشی بسیار مهم و حیاتی در نحوه‌ی مدیریت منابع سخت‌افزاری و نرم‌افزاری می‌باشد. پرداختن به مقوله‌ی امنیت سیستم‌های عامل، همواره از بحث‌های مهم در رابطه با ایمن‌سازی اطلاعات در یک سیستم کامپیوتری بوده است که امروزه با گسترش اینترنت، اهمیت آن دوچندان شده است. امروزه در دنیایی متکی بر فناوری اطلاعات زندگی می‌کنیم که هر آن، به خطر افتادن جریان اطلاعات منجر به بروز خسارت‌های تجاری جبران‌ناپذیری خواهد شد. تمامی شرکت‌ها و متخصصان امنیتی به‌دنبال یک سکوی<sup>1</sup> امن‌تر برای اجرای برنامه‌های کاربردی و سرویس‌دهنده‌ها هستند. لینوکس، حرف‌های بسیاری برای گفتن در سمت امنیت دارد. بسیاری از قابلیت‌های امنیتی که در سیستم عامل ویندوز وجود ندارند و یا تنها با افزودن نرم‌افزارهای اضافی قابل دسترسی می‌باشند، به‌طور درونی و پیش‌فرض در سیستم‌های عامل لینوکس و یونیکس پیاده‌سازی شده‌اند. لینوکس از ابتدا برای محیط‌های شبکه‌ای و چند کاربره طراحی شده است و همین باعث رعایت مسائل امنیتی از ابتدا در آن شده است، درحالی‌که سیستم عامل ویندوز اینگونه نبوده و هم‌اینک نیز از نظر امنیتی دارای کاستی‌های فراوانی است. برای نمونه، یک برنامه‌ی مخرب با استفاده از همین ضعف‌های امنیتی می‌تواند کل سیستم عامل را نابود کند، اما چنانچه مورد مشابهی در لینوکس وجود داشته باشد، حداکثر به دایرکتوری خانگی کاربر اجراکننده آسیب خواهد رسید و کل سیستم عامل از خطر در امان می‌ماند.

این‌گونه نیست که لینوکس فاقد هرگونه اشکال امنیتی باشد، بلکه باز بودن کد منبع آن باعث می‌شود تا بسیاری از اشکالات امنیتی پیش از ایجاد خسارت و در مراحل توسعه و برنامه‌نویسی برنامه بر ملا شده و رفع شوند. اگر اشکالی نیز در برنامه‌های منتشر شده یافت شود، به‌دلیل موجود بودن کد منبع، به‌سرعت برطرف می‌گردد؛ هرچند، در سیستم عامل ویندوز، شما باید منتظر شرکت مایکروسافت بمانید. سیستم عامل ویندوز دارای اشکال‌های امنیتی بسیاری است که به آسانی هم کشف نمی‌شوند و هنگامی کشف می‌شوند که خسارات جبران‌ناپذیری در اثر حمله از طریق آن ضعف‌های امنیتی رخ دهد که نمونه‌های آن را شاهد هستیم. می‌توان ادعا کرد که شمار بسیار کمی ویروس و برنامه‌ی مخرب برای لینوکس وجود دارد و این درحالی است که سالیانه بیش از هزار ویروس و برنامه‌ی مخرب گوناگون برای سیستم عامل ویندوز ایجاد می‌شود. این موضوع به‌خاطر گسترده نبودن لینوکس نیست (حدود 70 درصد از سایت‌های وب در جهان بر روی سیستم عامل لینوکس و سرویس‌دهنده‌ی وب آپاچی در حال اجرا هستند)، بلکه به‌دلیل وجود حفره‌های امنیتی بی‌شمار ویندوز و سیاست انحصارگرایی شرکت مایکروسافت است. این شرکت طوری رفتار و سیاست‌گذاری کرده است که مشتریان خود را تنها به محصولات خود عادت دهد.

بسیاری از کاربران ویندوز از اینترنت اکسپلورر و ابزار Outlook برای مرور وب و پست‌الکترونیک استفاده می‌کنند. یک ویروس‌نویس، می‌داند که اگر ویروسی را برای کاربران ویندوز بنویسد، بر روی کامپیوترهای 90 درصد آنها اثر خواهد کرد، زیرا بیشترشان از مرورگر IE و ابزار پست‌الکترونیک Outlook استفاده می‌کنند، اما در سیستم عامل لینوکس و یونیکس، شما دامنه‌ی گسترده‌ای از انتخاب‌ها و اجبار نداشتن در آنها در اختیار دارید. برای نمونه، برخی از کاربران لینوکس از مرورگر Mozilla Firefox استفاده می‌کنند، اما برخی مرورگر Konqueror را ترجیح می‌دهند. همچنین درخصوص ابزارهای پست‌الکترونیک از Kmail، Evolution، Pine، Mutt و یا Mozilla Mail استفاده می‌شود. در نتیجه

<sup>1</sup> Platform

فرد ویروس‌نویس تنها می‌تواند برای یکی از این ابزارها ویروس بنویسد و کد وی بر روی مابقی ابزارها عمل نخواهد کرد و عملاً میزان اثر آن اندک خواهد بود. ضمناً هیچ‌یک از ویروس‌هایی که برای سیستم‌عامل ویندوز نوشته شده‌اند، بر روی لینوکس کار نمی‌کنند. البته در مورد کدهای مخرب دو زیست، موضوع کمی متفاوت است. نخستین و برجسته‌ترین تفاوت در اینجا است که سیستم‌های مبتنی بر یونیکس و لینوکس به معنای کاملاً واقعی سیستم‌هایی چندکاربره هستند و برای هر فایل به تنهایی یا یک دایرکتوری می‌توان سطوح دسترسی کاربران و گروه‌های کاربری را تعریف کرد و هر کاربر به‌صورت پیش‌فرض دارای یک محدوده‌ی امن اطلاعاتی از فایل‌های شخصی خانگی است.

موردی که در اینجا درباره‌ی کاربران و گروه‌های کاربری لازم به گفتن است این است که هر کاربر در سیستم یونیکس و لینوکس یک دایرکتوری شخصی به‌نام home خواهد داشت که همه‌ی اختیارات دسترسی فایل در آن برای وی مجاز بوده و می‌تواند در آن به ایجاد و حذف فایل یا عملیات دیگر بپردازد و هیچ‌یک از کاربران یا گروه‌های دیگر به‌جز کاربر ریشه‌ی root در حالت عادی و بدون اجازه‌ی وی نخواهند توانست به اطلاعات او دسترسی یابند. در سیستم‌های مبتنی بر یونیکس و لینوکس، هر کاربری که مالک یا ایجادکننده‌ی یک فایل یا دایرکتوری باشد خواهد توانست با تعیین سطوح دسترسی، فایل یا دایرکتوری مربوط را برای دیگر کاربران یا گروه‌های کاربری آنها، از امکان خواندن و ایجاد تغییر و یا اجرای فایل اجرایی محروم یا بهره‌مند سازد. بدیهی است که مجوزهای تعریف شده برای یک گروه کاربری بر روی همه‌ی اعضای آن گروه اعمال می‌گردد و هر کاربر عضو آن گروه از تمام مزایا یا محدودیت‌های موصوف بهره‌مند خواهند شد.

موضوع تفاوت‌های لینوکس و ویندوز از جمله مواردی است که بسیاری از کاربران را بر سر دو راهی قرار داده است که کدام‌یک از این دو سیستم‌عامل می‌تواند خواسته‌های آنها را برآورده سازد. در این تفاوت‌ها به‌ویژه در حوزه‌ی امنیت سیستم‌عامل، کار را به دقت وافر رهنمون می‌شود، چراکه با توجه به وجود ویروس‌ها و نفوذگران مختلف و حتی وجود اشکال‌های سخت‌افزاری و نوسان‌های برق، نمی‌توان به سادگی از کنار موضوع امنیت سیستم‌عامل گذشت. آزاد و در دسترس بودن کدهای منبع سبب می‌شود تا بتوان از طرز کارکرد دقیق سیستم‌عامل آگاه شد؛ ویژگی که در سیستم‌عامل ویندوز به سادگی در اختیار کاربر قرار ندارد. امنیت برای کاربران عادی نیز در این سیستم‌عامل بسیار کارآمد خواهد بود. برای نمونه، وقتی چندین نفر از یک کامپیوتر استفاده می‌کنند، هر کدام می‌توانند تنها به اطلاعات خود دسترسی داشته باشند و نمی‌توانند برای اطلاعات دیگر کاربران تهدیدی به‌شمار آیند. خبری از کرم‌های اینترنتی، اسب‌های تروا و دیگر خرابکارها نیست.

## 1-1 مکانیزم‌های مختلف در بهبود امنیت لینوکس

بررسی‌های گوناگونی در مورد توزیع‌های مختلف لینوکسی انجام شده است و معمولاً از این سو و آن سو شنیده می‌شود که فلان توزیع از نظر امنیتی بهتر از توزیع دیگر است. از این رو بهتر است امنیت سیستم‌عامل لینوکس را از جنبه‌های مختلف بررسی کنیم. شاید نخستین و اصلی‌ترین بحث امنیت در سیستم، بحث تأمین امنیت فضای کاربر است. حملاتی که به سیستم‌های عامل انجام می‌شوند، به‌طور کلی حافظه و مشکلات مدیریت آن را هدف می‌گیرند و در واقع، تخریب حافظه یکی از راه‌های معمول برای نفوذ به سیستم‌های کامپیوتری مدرن است. ماجرا از این قرار است که یک سرریزی بافر می‌تواند کار را به حملات پیچیده‌ی تخریب حافظه بکشاند و نظام یکپارچه‌ی پروسه‌ها و حافظه‌ی سیستم را به هم بزند. امنیت سیستم‌ها هر چند که در طول سالیان مختلف و از هر نگرش به نگرش بعدی بهبود یافته است، اما تکنیک‌های هجوم به این بخش از سیستم در طول زمان پیچیده‌تر شده و در توزیع‌های مختلف لینوکس برای آزمون امنیت به‌کار می‌رود. بسیاری از کارشناسان معتقدند برخی از مکانیزم‌های امنیت در مقایسه با

دیگر روش‌ها، مقاوم‌تر هستند. افزون بر این، شماری از این روش‌ها تأثیر زیادی روی بازدهی سیستم می‌گذارند و حتی می‌توانند هنگام کامپایل نرم‌افزاری خاص، از نظر تطبیق با سیستم مشکل‌ساز شوند.

می‌توان اشاره کرد که این دو فاکتور، یعنی تأثیر در بازدهی سیستم و امکان از دست رفتن کنترل روی نگاه‌داری سیستم، بزرگ‌ترین دغدغه‌هایی است که تمام مکانیزم‌ها و روش‌های مقاومت در برابر حفره‌های امنیتی در توزیع‌های لینوکس با آن مواجهند. در هر نرم‌افزار و توسعه‌ی نرم‌افزاری آن می‌توان بدون هیچ تعجیبی دید که این مسئله حتی می‌تواند بحث‌های شدیدتری بین چندین پروژه به‌وجود بیاورد. مکانیزم‌های امنیتی که برای امن‌سازی باید مد نظر قرار گیرد، موارد زیر را دربر می‌گیرد:

- نگاشت حافظه‌ی غیراجرایی (استفاده از بیت سخت‌افزاری NX)
- کد یا برنامه‌ی مستقل از محل (PIC/PIE)
- پیاده‌سازی قابلیت FORTIFY\_SOURCE از glibc
- محافظت از نابودی پشته (SSP)
- RELRO یا محکم کردن برنامه‌های ELF با مرتب‌سازی بخش‌هایی از فایل و تبدیل آنها به حالت فقط‌خواندنی

همچنین ابزارهای زیر برای مقایسه استفاده شده است:

- ابزار checksec.sh، با اصلاح کوچکی که وضعیت FORTIFY\_SOURCE را نشان دهد.
- اسکریپت پایتونی که قابلیت‌های checksec.sh را افزایش می‌دهد.

هسته‌ی لینوکس، بیت NX را از زمان توزیع 2/6/8 پشتیبانی کرده است. از این‌رو برای سخت‌افزارهای 32 و 64 بیتی که این بیت را در مدار خود دارند، فعال است. برای بهره‌بردن از نگاشت حافظه‌ی غیراجرایی در سخت‌افزارهایی که از این قابلیت استفاده نمی‌کنند، لازم است که از وصله‌ی grsecurity یا بسته‌ی Exec Shield استفاده کرد. همچنین باید اشاره شود که بدون محافظت اضافه، مهاجم به‌سادگی می‌تواند بیت NX را دور بزند و دسترسی‌ها را به‌صورت دستی تنظیم کند. همچنین باید اشاره شود که به‌جز بیت NX، تمام مکانیزم‌های امنیتی که در این بخش به آن اشاره شده است در بخش فضای کاربری حافظه پیاده‌سازی شده‌اند و در سطح هسته نیستند. همچنین مهم است بدانیم حفره‌ای که در هسته‌ی لینوکس وجود دارد، ورای از مکانیزم‌های امنیتی که برای پردازش‌های فضای کاربر فعال شده است، می‌تواند مورد تهاجم قرار گیرد. از این‌رو می‌توان هسته‌های سیستم‌عامل امروزی را با پشتیبانی نابودی پشته کامپایل کرد، هر چند بهبود امنیت هسته، اغلب به نصب وصله‌های زیادی از جمله وصله‌ی grsecurity می‌انجامد. برخی از توزیع‌ها برای خود وصله‌های خاص امنیتی منتشر می‌کنند.

- RELRO: حفاظت ساختمان داده‌ی داخلی یک فایل ELF بر دوش RELRO است تا نگذارد مهاجم، اطلاعات آن‌را مشاهده کند یا جریان اجرای آن‌را کنترل نماید. این امنیت را با اصلاح جدول اتصال روندها (PLT) یا جدول عمومی آفست‌ها (GOT) که بخشی از فایل ELF است، تضمین می‌کنند. زمانی که از RELRO استفاده می‌شود، کل بخش GOT به حالت فقط‌خواندنی تغییر پیدا می‌کند و از این‌رو وقتی پردازش اجرا می‌شود، نمی‌تواند توسط هیچ مهاجمی قابل تغییر باشد. اگر RELRO به‌صورت جزئی پیاده‌سازی شود، تنها PLT و GOT به‌عنوان فقط‌خواندنی علامت زده می‌شوند. هر دوی این گزینه‌ها البته ساختمان داده‌ی ELF را از نو مرتب می‌کنند تا مهاجم نتواند این بخش‌ها را با بخش‌های دیگر ELF روی هم بیاندازد. همچنین گفتنی است که RELRO کاملاً نیازمند آن است که تمامی سمبل‌ها هنگام اجرای برنامه شناسایی شود تا بتوان کل GOT را به‌صورت فقط‌خواندنی در آورد. از این‌رو

برنامه‌های بزرگ‌تر با کندی بیشتری هنگام باز شدن روبه‌رو می‌شوند و تعداد زیادی از سمبل‌ها از کتابخانه‌های اشتراک یافته باید بارگذاری شود.

- محافظ خرابی پشته: محافظ خرابی پشته، سوءاستفاده از سرریزی بافر را با پیاده‌سازی بررسی‌های امنیتی بیشتر در پردازش پشته دشوارتر می‌کند. افزونه‌ی SSP در کامپایلر GCC از زمان نگارش 4/1 به این برنامه اضافه شد.

## 2-1 بهره‌گیری از خطاها برای امن‌سازی لینوکس

در این بخش به بررسی خطاهایی می‌پردازیم که در صورت رفع آنها امنیت بیشتری در سیستم‌عامل لینوکس مورد نظر ایجاد می‌گردد. این خطاها شامل موارد زیر می‌باشند:

انتخاب گذرواژه‌ی ساده و یا گذرواژه‌های پیش‌فرض: با توجه به سریع شدن پردازنده‌ها و امکان دسترسی به نرم‌افزارهایی که گذرواژه‌ها را کشف می‌کنند، حتی با انتخاب گذرواژه‌های پیچیده نیز، می‌توان آنها را شکست. با استفاده از ابزارهایی که در سیستم‌عامل یونیکس و لینوکس پیش‌بینی شده است مسئول سیستم می‌تواند اجازه‌ی تولید گذرواژه‌ها و دیگر مسائل مرتبط را کنترل نماید. در برخی از سیستم‌های عامل یونیکس، فایل‌ی با نام passwd در شاخه‌ی /etc/default وجود دارد که راهبر یونیکس می‌تواند با ایجاد تغییراتی در آن به کاربر اجازه ندهد که گذرواژه‌های ساده را انتخاب نماید، اما در لینوکس به اندازه‌ی کافی کنترل بر روی گذرواژه‌ها انجام می‌شود و می‌توان تا حدی مطمئن بود که کاربر نمی‌تواند گذرواژه‌های ساده را انتخاب کند. فراموش نگردد که مسئول سایت (راهبر سیستم) این اختیار را دارد که گذرواژه‌های ساده‌ای را برای کاربران تهیه نماید، که این کار خطای مسلم راهبر می‌باشد. چراکه هر گذرواژه‌ی ساده، دروازه‌ای برای ورود افراد مهاجم بوده و فرد مهاجم پس از وارد شدن به سیستم می‌تواند با استفاده از نقاط ضعف احتمالی دیگر و به‌وجود آوردن سرریز بافر<sup>1</sup> کنترل سیستم را در دست بگیرد. در بسیاری از سیستم‌های فعلی یونیکس و لینوکس، مجموعه امکانات Pluggable Authentication Modules نصب می‌باشد و توصیه‌ی اکید می‌گردد که کد زیر برای بالا بردن امنیت سیستم، تحت /etc/pam.d و در فایل passwd قرار گیرد:

```
passwd password require /lib/security/pamcracklib.so retry=3
passwd password require /usr/lib/security/pam_pwdb.so use_authtok
```

در زمان اجرای برنامه‌ی passwd، کتابخانه‌های پویا با نام‌های pamcracklib.so و pam\_pwdb.so به برنامه متصل شده و کنترل‌های لازم را انجام می‌دهند. مجموعه نرم‌افزارهای cracklib این امکان را به سیستم اضافه می‌کند تا کنترل نماید که آیا گذرواژه‌ی تهیه شده توسط کاربر شکستی است یا خیر. فراموش نشود که فرمان passwd تابع راهبر سیستم می‌باشد و راهبر سیستم می‌تواند گذرواژه‌ی ساده را انتخاب نماید و این عمل گناهی نابخشودنی را برای مسئول سیستم ثبت خواهد کرد. در مورد گذرواژه‌های پیش‌فرض که در هنگام نصب برخی سوئیچ‌ها و مسیریاب‌ها وجود دارد، راهبر سیستم باید به سرعت (زمان نصب گذرواژه‌های از پیش تعیین شده را تعویض نماید).

- باز گذاشتن پورت‌های شبکه: باز گذاشتن پورت‌هایی که محافظت نشده و یا بدون استفاده می‌باشند، به مهاجمان اجازه می‌دهد به‌نحوی وارد سیستم شوند و امنیت سیستم را مخدوش نمایند. فرمان‌های زیادی مانند finger و rwho و غیره وجود دارند که افراد مهاجم می‌توانند با اجرای آنها در شبکه و قرار دادن آدرس کامپیوتر مقصد، نام‌های کاربران و تعداد زیادی از قلم‌های اطلاعاتی مربوط به کاربران را به‌دست آورند و با حدس زدن گذرواژه

<sup>1</sup> Buffer Overflow

وارد سیستم شوند. به‌وسیله‌ی ابزارهایی که در سیستم‌عامل یونیکس و لینوکس وجود دارد می‌توان پورت‌های باز را پیدا کرد و تمهیدات لازم را انجام داد. یکی از این ابزارها، NMap نام دارد که با اجرای این فرمان و درج سوئیچ‌ها و گزینه‌های لازم و وارد کردن آدرس IP، پورت‌های کامپیوتر مورد نظر را یافت و فعالیت‌های اختلال‌گونه را انجام داد. راهبر سیستم با اجرای فرمان `netstat -atuv` می‌تواند سرویس‌هایی که در حال اجرا هستند را مشخص کند و با انواع روش‌هایی که وجود دارد، سرویس را غیرفعال نماید و شاید یک روش مناسب برای پاک کردن برنامه‌های سرویس‌دهنده و یا تغییر مجوز آن (با فرمان `chmod`) باشد. هرچند، می‌توان با فرمان `chkconfig` اجرای برخی از سرویس‌ها را در زمان راه‌اندازی سیستم متوقف کرد. برای نمونه، با کمک فرمان `chkconfig` به‌فرم زیر می‌توان سرویس `portmap` را بر روی سیستم مورد نظر غیرفعال نمود:

```
# chkconfig -del portmap
```

- استفاده از نرم‌افزارهای قدیمی: توصیه می‌شود که از نرم‌افزارهایی که نسخه‌های جدید آن به‌دلیل وجود اشکالات امنیتی در نسخه‌های قدیمی روانه‌ی بازار شده است، استفاده شود و گناهی بس نابخشودنی است که راهبر سیستم با استفاده از نرم‌افزارهای قدیمی، راه را برای سوءاستفاده‌کنندگان باز بگذارد. برای نمونه، فرمان `ls` دارای مشکلی است که با قرار دادن آرگومانی خاص می‌توان سرریز بافر را به‌وجود آورد و کنترل سیستم را به‌دست گرفت. چندی پیش مجموعه نرم‌افزارهای مربوط به نمایش نام فایل‌ها و شاخه‌ها (`ls`، `dx`، `lr` و غیره) در سایت‌های مهم و استاندارد قرار داده شد تا استفاده‌کنندگان لینوکس آن‌را بر روی سیستم خود نصب نمایند.
- استفاده از برنامه‌های نا امن و یا پیکربندی شده به‌صورت نادرست: به‌دلیل مسائل خاص، برخی از سیستم‌ها نیاز به مجوزهای ویژه دارند و اعمال مجوزها می‌تواند مسائل غیرقابل پیش‌بینی را به وجود آورد و ضمناً با پیکربندی نامناسب نرم‌افزار، راه برای سوءاستفاده‌کنندگان باز خواهد شد. برای نمونه، نرم‌افزارهایی وجود دارند که برای اجرا شدن، مجوز `Set UserID` را لازم داشته (مجوز S) و این مجوز درحالتی که صاحب فایل اجرایی `root` باشد، بسیار خطرناک است. فرمانی که این اجازه را دارد با اجرای فراخوان‌های سیستم<sup>1</sup> مانند `setuid` تغییر مالکیت داده و قدرت `root` را کسب می‌کند و راهبر سیستم باید تاوان این گناه نابخشودنی را نیز پس بدهد. برای نمونه، استفاده از `FTP` و `telnet` که اطلاعات را عیناً بر روی شبکه منتقل می‌کنند، می‌تواند نگرانی‌هایی را برای مسئول سایت به وجود آورد و شاید راه‌اندازی `sshd` بتواند کمی از گناهان مسئول سیستم را بکاهد و در مورد پیکربندی نادرست فایل‌ها بتوان نامی از فایل `rhosts`. برد که مجوز نادرست می‌تواند باعث لو رفتن گذرواژه گردد. بهتر است با فرمان `find` نام فایل‌هایی که مجوز `s` را دارند، کنترل کنید تا برنامه‌ی اجرایی با مجوز `s` در سیستم اضافه نگردد. همچنین، مسئول سیستم در اجرای دستور `mount` نیز باید دقت فراوان داشته باشد تا برنامه‌هایی که مجوز `s` بر روی `CD/DVD` و فلاپی دارند، اجرا نشود.
- ناکافی بودن منابع و یا نامناسب بودن ارجحیت‌ها: کم کردن هزینه‌های مربوط به امنیت و عدم آموزش‌های لازم و تهیه نکردن نرم‌افزارهای بازدارنده می‌تواند شماری مسائل غیرقابل پیش‌بینی به وجود آورد. به‌ویژه جابه‌جایی اولویت‌های هزینه کردن اعتبارات می‌تواند امنیت سیستم را خدشه‌دار نماید. لازم به یادآوری است که این مطلب فنی نیست و مدیریتی می‌باشد، اما راهبر سیستم باید مرتباً نکات لازم را در این زمینه به مقامات مسئول گوشزد نماید تا مدیریت ارشد سازمان بیش از پیش به اهمیت امنیت پی ببرد و هزینه‌های لازم را تأمین نماید. عدم

<sup>1</sup>\_System call

اطلاع‌رسانی مسئول سایت در این زمینه به مدیریت‌های مافوق که احتمالاً در این زمینه تخصصی ندارند نیز اشتباهی بسیار بزرگ به‌شمار می‌رود.

- نگه‌داری UserID های قدیمی و غیر لازم و تهیه‌ی شناسه‌های عمومی: نگه‌داری UserID های قدیمی و شناسه‌هایی مانند TEST می‌تواند معضلات زیادی را به وجود آورد و امکان سوءاستفاده را بالا ببرد. تهیه‌ی شناسه‌های عمومی نیز به دلیل نامشخص بودن هویت اصلی کاربر می‌تواند مشکل‌زا باشد. مسئول سایت باید رویه‌ای را برای کشف UserID های غیرفعال اتخاذ نماید و با هر روشی که صلاح می‌داند پس از تهیه‌ی فایل پشتیبان لازم، UserID های غیرفعال را در مقاطع معینی متوقف نماید و شاید یکی از بهترین روش‌ها برای انجام این کار تعویض گذرواژه باشد. برای نمونه، با دستور زیر می‌توان UserID با نام someone را غیرفعال کرد:

```
# chmod 000 /home/someone
```

تولید UserID های عمومی مانند test و guest و غیره که مورد علاقه‌ی بسیاری از مهاجمان است، یکی دیگر از بزرگ‌ترین اشتباه‌های غیرقابل بخشش راهبر سیستم می‌باشد.

- به تعویق انداختن فعالیت‌های مهم در زمینه‌ی ایجاد امنیت: با کم اهمیت دادن مسائل حفاظتی همچون نصب نکردن ترمیم‌ها یا وصله‌ها<sup>1</sup> و تهیه نکردن فایل‌های پشتیبان، می‌توان گفت که مسئول سیستم تیر خلاص را به کامپیوترهای خود شلیک کرده است و خطایی بسیار بزرگ را مرتکب شده است.

### 3-1 توصیه‌های مهم امنیتی در مورد یونیکس و لینوکس

بررسی و تجزیه و تحلیل امنیت در سیستم‌های عامل باید با ظرافت و در چارچوبی کاملاً علمی و با در نظر گرفتن تمامی واقعیت‌های موجود، انجام شود تا امکان نگه‌داری و پشتیبانی سیستم‌ها با در نظر گرفتن مجموعه تهدیدات موجود و آتی، به سرعت و به سادگی میسر گردد. بیشتر حمله‌های موفقیت‌آمیز در اینترنت، به دلیل وجود نقاط آسیب‌پذیر در شماری اندک از سرویس‌های سیستم‌های عامل متداول است. مهاجمان، با فرصت‌طلبی خاص خود از روش‌های فراوانی به منظور سوءاستفاده از نقاط ضعف امنیتی شناخته شده، استفاده می‌کنند و در این راستا ابزارهای متنوع، کارآمد و گسترده‌ای را به منظور نیل به اهداف خود، به خدمت می‌گیرند. مهاجمان، در این رهگذر متمرکز بر سازمان‌ها و مؤسسه‌هایی می‌گردند که هنوز مسائل موجود امنیتی (حفره‌ها و نقاط آسیب‌پذیر) خود را برطرف نکرده‌اند و بدون هیچ‌گونه تبعیضی آنها را به عنوان هدف، انتخاب می‌کنند. پس با شناسایی و تجزیه و تحلیل این گونه نقاط آسیب‌پذیر توسط کارشناسان امنیت اطلاعات، سازمان‌ها و مؤسسات قادر به استفاده از مستندات علمی تدوین شده به منظور برخورد منطقی با مشکلات موجود و ایجاد یک دیوار حفاظتی مناسب می‌باشند.

لینوکس و یونیکس، از سیستم‌های عامل رایج در جهان هستند که امروزه در سطح بسیار گسترده‌ای استفاده می‌گردند. تاکنون حمله‌های بی‌شماری توسط مهاجمان متوجه سیستم‌هایی بوده است که از یونیکس و لینوکس (نسخه‌های متفاوت) به عنوان سیستم‌عامل استفاده می‌کنند. با توجه به حمله‌های متنوع و گسترده‌ی انجام شده، باید سبک مقابله با این حمله‌ها و تهدیدها شناخته شود و راه‌های نفوذ به سرعت ترمیم گردد و کاملاً مسدود شود. عوامل فراوانی در بُروز این گونه حمله‌ها نقش دارد که از آن جمله می‌توان به آگاه نبودن لازم مدیران سیستم درباره‌ی ارتقاء امنیتی سیستم‌هایی که بر روی آنها نرم‌افزارهای مدیریت اطلاعات شبکه نصب یا به صورت غیرضروری اجراء می‌گردد و

<sup>1</sup> Patches

پیکربندی نامناسب برنامه‌ها اشاره کرد. این نمونه‌ها می‌تواند زمینه‌ی یک تهاجم از نوع DoS ویا یک سرریزی بافر را فراهم سازد. به‌منظور حفاظت سیستم و ترمیم سریع‌تر نقاط آسیب‌پذیر، موارد زیر پیشنهاد می‌گردد:

- همیشه آخرین نسخه‌ی نرم‌افزارهای ارائه شده را دریافت و آن‌را بر روی سیستم نصب نمایید. به‌منظور به‌هنگام‌سازی سیستم باید از تمامی وصله‌های ارائه شده توسط تولیدکنندگان استفاده شود و درصورت امکان آن نرم‌افزار را به آخرین نسخه‌ی موجود ارتقاء دهید.
- دیوارآتش موجود بر روی سیستم را دقیقاً نصب و با توجه به نیازهای امنیتی خود در سطح مناسب پیکربندی کنید.
- پورت‌های غیرضروری یا در معرض تهدید را در سطح مسیریاب ویا دیوارآتش، با توجه به توصیه‌های امنیتی موجود در بولتن‌های امنیتی مسدود کنید.
- آن دسته از نرم‌افزارهای غیرضروری که به‌صورت پیش‌فرض هنگام نصب سیستم‌های عامل نصب می‌گردند و هیچ‌گونه کاربرد عملی ندارند را غیرفعال نمایید.
- به‌منظور پیچیده‌تر کردن حمله‌های خودکار ویا جلوگیری از پویبش غیرمجاز سیستم توسط مهاجمان، به توصیه‌های امنیتی تهیه‌کنندگان نرم‌افزارها در سایت‌های مربوط یا دیگر سایت‌های امنیتی لینوکس و یونیکس دقیقاً عمل کنید و هیچ نکته‌ای را از قلم نیاندازید.
- اکیدا استفاده از حساب‌های کاربری با گذرواژه‌ی ضعیف ویا فاقد گذرواژه را مسدود کنید، زیرا گذرواژه دارای نقشی حیاتی و اساسی در ایجاد نخستین سطح دفاع در یک سیستم اطلاعاتی است و از دست‌رفتن آن ویا ضعف آن می‌تواند سیستم را در معرض تهدیدات جدی قرار دهد. مهاجمان پس از دستیابی به گذرواژه‌ی کاربران تأیید شده (استفاده از مکانیزم‌های متفاوت) قادر به دستیابی به منابع سیستم و حتی تغییر در تنظیمات دیگر حساب‌های کاربری تعریف شده و موجود بر روی سیستم خواهند بود، عملیاتی که می‌تواند پیامدهای بسیار منفی را به‌دنبال داشته باشد. چنانچه از حساب‌های کاربری استفاده می‌شود که بین کاربران متعدد ویا کارکنان موقت یک سازمان به اشتراک گذاشته شده است ویا کاربران از رمزهای عبور به‌درستی حفاظت ننمایند، پتانسیل نفوذ به شبکه توسط یک مهاجم فراهم می‌گردد.
- از یک برنامه‌ی پویبش‌گر به‌روز شده که قادر به بررسی دقیق سیستم‌های کامپیوتری به‌منظور تشخیص نقاط آسیب‌پذیر باشد، استفاده کنید. با عضویت در گروه‌های خبری چون Symantec برای آگاهی از آخرین هشدارهای امنیتی اطلاعات، خود را به‌روز نگاه دارید.
- از امکانات رمزنگاری چون OpenSSH در سرویس‌های شبکه مانند SMTP، Telnet، POP3، IMAP، rlogin، HTTP و غیره استفاده کنید.

#### 4-1 مفاهیم پایه‌ای در امنیت سیستم‌های عامل کدباز (لینوکس و یونیکس)

در زمینه‌ی امنیت لینوکس و مقایسه‌ی آن با سیستم‌های فراگیر ویندوز، حرف و حدیث‌های بسیاری مطرح است. براساس آمارهای شرکت‌های امنیتی، بدافزارهای بسیاری برای سیستم‌های عامل سری ویندوز ساخته شده و گسترش یافته‌اند، درحالی‌که سیستم‌عامل لینوکس تنها هدف تعداد بسیار محدودی از بدافزارها قرار گرفته است. با این اوصاف،

بسیاری معتقدند، بر مبنای اصل امنیت از طریق ابهام<sup>1</sup>، سیستم‌های منبع بسته امنیت بیشتری دارند و به دلیل این که کد آنها در اختیار همگان نیست، مشکلات امنیتی آنها نیز کمتر مشهود بوده و امنیت بیشتری دارند. در اینجا باید گفت، این ادعا کاملاً اشتباه است و ادعایی کاملاً نادرست می‌باشد. با اینکه کد سیستم‌عامل لینوکس و برنامه‌های آن باز است و در اختیار همگان قرار دارد، اما فضای اجرایی نرم‌افزار در سیستم‌عامل لینوکس و برنامه‌های آن باز بودن کد برنامه‌ها به محض نمایان شدن حفره‌های امنیتی در سیستم، هر کسی حتی خود کاربر می‌تواند به برطرف کردن این عیوب اقدام کند و این امر سرعت پاسخ به مشکلات امنیتی را بسیار افزایش می‌دهد. حال این قابلیت و امکانات را با ویندوز و سیستم‌های زیرمجموعه‌ی منبع بسته‌ای مقایسه کنید که در برخی از موارد، ارائه‌ی اصلاحیه‌ی امنیتی برای سیستم تا یک ماه پس از آشکار شدن مشکل طول می‌کشد. در ادامه‌ی این بخش به بررسی برخی از مفاهیم پایه‌ای امنیت در سیستم‌های عامل یونیکس و لینوکس می‌پردازیم:

#### 1-4-1 مفهوم TCPwrapper

TCPwrapper یا tcpd به صورت واسطه‌ای بین دایمون inetd و سرویسی که inetd فراخوانی می‌کند، اجرا می‌شود. می‌توان TCPwrapperها را برای فیلتر کردن و نیز ثبت وقایع اتصال‌ها به‌ازای هر سرویس تنظیم کرد. همان گونه که می‌دانید برای محافظت سرورها می‌توان از دیوارهای آتش استفاده کرد. البته در برخی موارد نیاز به محافظت از سیستم توسط TCPwrapper نیز خواهیم داشت. TCPwrapper نیز همانند دیگر سرویس‌های لینوکسی دارای فایل‌های ویژه‌ی پیکربندی است. دو فایل که با این سرویس استفاده می‌شود عبارت‌اند از:

/etc/hosts.allow , /etc/hosts.deny

زمانی که اتصالی به یک پورت خاص برقرار می‌شود، دایمون inetd برای بررسی مجاز بودن درخواست طبق قواعد موجود در فایل‌های مذکور، آن را برای دایمون tcpd ارسال می‌کند. در صورت مجاز بودن، tcpd درخواست را برای سرویس مورد نظر ارسال می‌کند، اما در صورتی که اتصال غیرمجاز باشد، درخواست مربوط متوقف و مسدود می‌شود. اگر از سیستم لینوکسی خود تنها به صورت یک سیستم خانگی استفاده می‌کنید و نیازی به دسترسی از راه دور به آن ندارید، پیشنهاد می‌شود، همه‌ی اتصال‌ها از راه دور را با تنظیمات زیر مسدود کنید. برای مسدودسازی همه‌ی اتصال‌ها از تمام میزبانان راه دور، سطر زیر را به فایل /etc/hosts.deny بیافزایید:

ALL: ALL

همچنین برای پذیرفتن همه‌ی اتصال‌ها از میزبانان محلی، سطر زیر را به فایل /etc/hosts.allow بیافزایید:

ALL: LOCAL

اکنون فرض کنید قصد داریم اجازه‌ی دسترسی از نوع SSH را برای کلاینت‌های cli1، cli2 و cli3 صادر نماییم. برای انجام این کار باید سطر زیر را به فایل /etc/hosts.allow بیافزاییم:

sshd: cli1 cli2 cli3

اینک فرض کنید قصد داریم سیستم‌های موجود در یک محدوده از آدرس IP، مجوز دسترسی از نوع SSH را داشته باشند. در این حالت به فرم زیر عمل می‌کنیم:

sshd: cli1 cli2 cli3 . subnet. example.com

<sup>1</sup> Security via Obscurity



چنانچه قصد دارید تمام سیستم‌های یک محدوده از آدرس IP مجوز اتصال داشته باشند، اما تنها یک سیستم از این محدوده این اجازه را نداشته باشد، به صورت زیر عمل کنید:

```
ALL: . subnet. example.com EXCEPT exam. subnet. example.com
```

بر اساس دستور بالا همه‌ی سیستم‌ها به جز سیستم exam، مجوز دسترسی خواهند داشت. دقت داشته باشید که فایل hosts.allow بر hosts.deny حق تقدم دارد. ضمناً این دو فایل قابل خواندن برای همه‌ی کاربران می‌باشند که البته می‌توانید مجوز دسترسی خواندن را از آنها بردارید. با وجود اینکه امکان استفاده از TCPwrapper در کنار xinetd وجود دارد، اما در حالت کلی اهمیت ندارد. دایمون xinetd خصیصه‌های کنترل دسترسی و ثبت وقایع TCPwrapper را با یکدیگر ترکیب می‌کند. نسخه‌ای از TCPwrapper که همراه توزیع Red Hat ارائه می‌شود، گزینه‌های بسیار پیچیده‌تری نیز دارد که قابلیت‌های فوق‌العاده‌ای به آن داده است.

## 2-4-1 مفهوم دایمون xinetd

xinetd<sup>1</sup>، جایگزین نسل جدید برای دایمون inetd است. این سرویس افزون بر توانایی کنترل دسترسی همانند TCPwrapper، انعطاف‌پذیری بیشتری در تنظیمات و ثبت وقایع دارد. xinetd که همراه بیشتر توزیع‌های لینوکس ارائه می‌شود خود شامل یک TCPwrapper است که در داخل دایمون xinetd گنجانده شده است و به سادگی می‌توان از آن برای فعال یا غیرفعال کردن سرویس‌های شبکه‌ای استفاده کرد. افزون بر فایل اصلی تنظیمات که در مسیر /etc/xinetd.conf جای دارد، به ازای هر سرویس، فایل‌های دیگری برای تنظیمات وجود دارد که در شاخه‌ی /etc/xinetd.d/ جای دارند. همان‌گونه که گفته شد، سرویس xinetd جایگزینی برای inetd می‌باشد که همه‌ی سرویس‌های شبکه‌ای را که روی xinetd تنظیم شده‌اند، کنترل و نظارت می‌کند. هم‌اینک برخی از سیستم‌های عامل مبتنی بر یونیکس مانند FreeBSD، برای پیکربندی سرویس‌های شبکه‌ای خود از دایمون inetd استفاده می‌کنند. با کمک دستور زیر می‌توانید از فعال بودن دایمون xinetd بر روی سیستم، اطمینان حاصل نمایید:

```
# chkconfig --list xinetd
```

در صورتی که قصد دارید شماره فرایند آن را نیز مشاهده کنید، از دستور زیر استفاده نمایید:

```
# /etc/init.d/xinetd status
```

اگر xinetd بر روی سیستم فعال باشد، باید بدانید که چه سرویس‌هایی با این دایمون در حال فعالیت هستند. گفتنی است که به دایمون xinetd، در اصطلاح Super Server نیز گفته می‌شود. با کمک دستور زیر می‌توان فهمید که چه سرویس‌هایی تحت دایمون xinetd پیکربندی شده‌اند و همچنین می‌توان فهمید که پورت‌های کدام‌یک از آنها توسط دایمون xinetd کنترل و نظارت می‌شود:

```
# chkconfig --list | awk '/xinetd based services/,/"/'
```

اینک اگر تنها قصد دیدن سرویس‌هایی را دارید که پورت‌های آنها با دایمون xinetd کنترل و نظارت می‌شود، از دستور زیر استفاده نمایید:

```
# chkconfig --list | awk '/xinetd based services/,/"/' | grep -v off
```

فرض کنید به دلیل وجود آسیب‌پذیری‌های امنیتی موجود در سرویس Telnet، قصد دارید این سرویس را بر روی یک سرور غیرفعال نمایید. گفتنی است که این سرویس نیز در قالب دایمون xinetd می‌باشد. برای انجام این کار ابتدا باید

<sup>1</sup> Extended Internet Services Daemon

بررسی کنید که آیا این سرویس بر روی سرور مربوط فعال است یا خیر. برای این منظور می‌توانید از یکی از دستورات زیر استفاده کنید:

```
# chkconfig --list telnet
```

```
# cat /etc/xinetd.d/telnet | grep disable
```

با کمک دستور دوم یک سطر از فایل پیکربندی مخصوص سرویس Telnet را مشاهده خواهید کرد که نشان می‌دهد این سرویس تحت xinetd فعال است. در ادامه با کمک دستور زیر سرویس Telnet را غیرفعال می‌کنیم:

```
# chkconfig telnet off
```

برای اطمینان بیشتر بهتر است سرور Telnet را به‌طور کامل از روی سرورها و سیستم‌های حساس حذف کنید و به‌جای آن از سرویس SSH استفاده نمایید:

```
# rpm -e telnet-server
```

اینک فرض کنید ممکن است یک سرویس تحت xinetd داشته باشید که اطلاعاتی در مورد آن در اختیار ندارید و نمی‌دانید مورد استفاده‌ی آن چیست. برای اینکه بتوانید در مورد یک سرویس خاص اطلاعات کاملی گردآوری کنید و لزوم وجود یا عدم وجود آن را مشخص سازید، به‌فرم زیر عمل کنید. فرض کنید با کمک دستور زیر به این نتیجه رسیدید که سرویسی با نام authd تحت دایمون xinetd کار می‌کند، اما نمی‌دانید چه سرویسی است و چه کاری در سیستم انجام می‌دهد:

```
# chkconfig --list xinetd
```

در این حالت ابتدا با کمک دستور زیر، مسیر دقیق دایمون سرویس مذکور را پیدا می‌کنیم:

```
# grep "server" /etc/xinetd.d/auth
```

خروجی دستور بالا به‌فرم زیر می‌باشد:

```
server = /usr/sbin/in.authd
```

```
server_args = -t60 --xerror --os -E
```

در ادامه باید با کمک دستور زیر یک دستور پرس و جو از مدیر بسته‌های نرم‌افزاری سیستم دریافت کنیم که ببینیم فایل in.authd در کدام بسته‌ی نرم‌افزاری جای دارد:

```
# rpm -qf /usr/sbin/in.authd
```

فرض کنید خروجی دستور بالا به‌فرم زیر باشد:

```
authd-<Version>. rhel3
```

اکنون باید با کمک دستور پرس و جو دیگری، توضیحات مربوط به این بسته‌ی نرم‌افزاری را به‌دست آوریم:

```
# rpm -qi authd-<Version>. rhel3 | awk '/Description/, /"/'
```

چنانچه قصد دارید همه‌ی فایل‌های موجود در این بسته‌ی نرم‌افزاری را نیز مشاهده کنید، دستور زیر را وارد نمایید:

```
# rpm -ql authd-<Version>. rhel3
```

اینک با توجه به توضیحات و اطلاعات به‌دست آمده درباره‌ی دایمون مذکور و با در نظر گرفتن سیاست‌گذاری سازمان مربوط می‌توانید در مورد فعال یا غیرفعال کردن سرویس مذکور تصمیم‌گیری کنید. همان‌گونه که می‌دانید برای غیرفعال کردن این سرویس باید به‌فرم زیر عمل کنید:

```
# chkconfig auth off
```

3-4-1 مفهوم دایمون `identd`

این سرویس برای احراز هویت به کار می‌رود و صاحب هر اتصال TCP/IP به یک سرور از راه دور، پذیرنده‌ی آن را شناسایی می‌کند. زمانی که کاربری از راه دور به یک میزبان متصل می‌شود، دایمون `inetd` بر روی سیستم میزبان راه دور، درخواستی روی پورت شماره 113 با عنوان "چه کسی دارنده‌ی این اتصال خاص به من است؟" برمی‌گرداند، `inetd` محلی نیز، پاسخ می‌دهد، مثلاً "نام کاربری Grayhat دارنده‌ی اتصال است". این سرویس نباید به‌عنوان روشی برای احراز هویت استفاده شود، زیرا هر کسی می‌تواند با داشتن دسترسی ریشه، پاسخ دایمون `inetd` را تغییر دهد. در واقع بر روی بیشتر سیستم‌های عامل مانند FreeBSD و ویندوز حتی یک کاربر معمولی نیز می‌تواند پاسخ دایمون `identd` را مطابق با چیزی که می‌خواهد، مشخص کند. این پروتکل روی سیستم‌های چندکاربره به‌عنوان روشی برای پیگیری کاربران مشکوک سودمند است. اگر یکی از کاربران شما مشکلی روی سیستم دیگری به‌وجود آورد، مدیر سیستم می‌تواند شما را از نام کاربری، کاربر مشکل‌زا با خبر کند. آیا لازم است شما سرویس `inetd` را اجرا کنید؟ بر روی سیستم‌هایی با کاربران زیاد، این کار فایده‌ی بسیاری دارد، اما هدف خاصی را روی یک سیستم تک‌کاربره دنبال نمی‌کند. در واقع اجرای آن به‌منزله‌ی باز گذاشتن سرویس برای دنیای خارج با تمام خطرات امنیتی است. نکته‌ی دیگری که باید در نظر گرفت این است که دایمون `identd` می‌تواند به نفوذگران اجازه دهد تا اطلاعات با ارزشی از سیستم شما به‌دست آورند. این اطلاعات می‌تواند شامل سرویس‌های خاصی باشند که با دسترسی ریشه در حال اجرا هستند. همچنین نوع سیستم‌عامل از طریق این دایمون برای نفوذگران قابل تشخیص می‌باشد. یکی از سوئیچ‌های این دایمون، گزینه‌ی `n` - آن می‌باشد که به‌جای نام کاربری، شماره کاربری را برای مقصد ارسال می‌کند. همین کار می‌تواند اطلاعات مفیدی در اختیار نفوذگر قرار دهد. برای آگاهی از گزینه‌های مختلف در اجرای این سرویس به فایل پیکربندی آن در مسیر زیر مراجعه نمایید:

```
/etc/identd.conf
```

در حالت کلی تأثیرات غیرفعال کردن دایمون `identd` شامل موارد زیر می‌باشند:

- میزبانان راه دور ممکن است برای پیگیری حمله‌هایی که از سایت شما نسبت به کاربران خاصی صورت می‌گیرد، دچار مشکل شوند. این مسئله زمانی اهمیت پیدا می‌کند که شما کاربران زیادی روی میزبان خود داشته باشید. غیرفعال کردن این سرویس روی سیستم چندکاربره، نه تنها یک عمل نادرست مدیریتی است، بلکه پیگیری کاربران مشکوک را در سایت شما مشکل خواهد کرد.
  - برخی از سرویس‌دهنده‌های FTP و بیشتر سرویس‌دهنده‌های IRC برای احراز هویت کاربران از راه دور به این سرویس نیاز دارند. غیرفعال کردن این سرویس تعداد سرویس‌دهنده‌های IRC و FTP که اتصال‌ها شما را می‌پذیرند، محدود می‌کند، اما چنانچه از IRC استفاده نمی‌کنید، مشکلی ایجاد نمی‌کند.
- می‌توانید دسترسی به سرویس `identd` را با علامت‌گذاری از نوع "# (Comment)" سطر `auth` در فایل `/etc/inetd.conf` یا با استفاده از `TCPwrapper` ها و یا نرم‌افزار دیوارآتش، محدود یا غیرفعال کنید. اگر برای اتصال به یک سرور خاص، نیاز دارید سرویس `identd` فعال باشد، این نکته را در نظر بگیرید که تنها به دسترسی‌هایی که از طرف آن سرور صورت می‌گیرد، مجوز دهید. اگر پورت مربوط به `identd` را توسط دیوارآتش کنترل می‌کنید، حتماً تلاش کنید از سیاست `Reject` به‌جای `deny` استفاده کنید. استفاده از `deny` ممکن است زمان اتصال شما به سرورهایی که از `identd` استفاده می‌کنند را به شدت افزایش دهد، زیرا پیش از برقراری اتصال، منتظر پاسخ می‌مانند.

## 5-1 مقاوم‌سازی سرورهای لینوکسی

امن‌سازی سرورهای لینوکس برای محافظت از داده‌های کاربران و مشتریان امری حیاتی و عقلانی است و تأمین امنیت سرورهای لینوکسی، یکی از وظایف مدیران سیستم است. در این بخش راه‌کارهایی برای افزایش امنیت یک سرور لینوکسی تازه نصب شده را عنوان می‌کنیم. این مثال را با یک سرور مبتنی بر Red Hat انجام می‌دهیم، اما برای نصب یا حذف بسته‌ها می‌توانید از ابزار مدیریت بسته‌ی توزیع خود استفاده کنید.

▪ **رمزگذاری ارتباطات داده:** تمام داده‌های ارسال شده از طریق شبکه قابل شنود است. برای جلوگیری از شنود داده‌ها، داده‌های ارسال شده را تا حد ممکن با گذرواژه یا استفاده از کلیدها و گواهی‌نامه‌های امنیتی رمزگذاری کنید. برای انجام این کار موارد زیر پیشنهاد می‌گردد:

➤ از سرویس‌های scp، SSH، rsync و SFTP برای ارسال داده‌ها استفاده کنید. همچنین می‌توانید سیستم‌فایل سرور راه دور یا پوشه‌خانگی کامپیوتر سرور را با استفاده از ابزار ویژه‌ی sshfs یا ابزار fuse به سیستم بیافزایید و سپس به انتقال داده بپردازید.

➤ برای شما امکان رمزگذاری و امضا کردن داده‌ها و ارتباطات را فراهم می‌کند. همچنین یک سیستم مدیریت کلید همه‌کاره و ماژول‌های دسترسی به انواع کلید عمومی را به‌همراه دارد.

➤ Fugu یک رابط‌گرافیکی برای برنامه‌ی انتقال امن فایل Sftp است. SFTP همانند ftp است، اما برخلاف آن، همه‌ی نشست را رمزگذاری می‌کند و این به آن معنا است که هیچ رمزعبوری به شکل متنی ارسال نخواهد شد. گزینه‌ی دیگر FileZilla است که یک کلاینت مستقل از سکو می‌باشد و از پروتکل FTP، FTPS یا SSL/TLS و پروتکل انتقال فایل ssh یا همان SFTP، پشتیبانی می‌کند.

➤ OpenVPN یک VPN سبک و کم هزینه با پشتیبانی SSL است.

➤ پیکربندی و نصب SSL Lighthttpd یا همان HTTPS را در نظر داشته باشید.

➤ پیکربندی و نصب Apache SSL یا HTTPS از طریق ماژول mod\_ssl آپاچی را نیز فراموش نکنید.

➤ در این باره از به‌کار بردن سرویس‌ها و پروتکل‌هایی چون FTP، Telnet و Rlogin/Rsh پرهیز کنید: در بیشتر پیکربندی‌های پیش‌فرض شبکه‌ها، نام‌های کاربری، گذرواژه‌ها، دستورهای FTP/Telnet/Rsh و فایل‌های انتقال یافته به سادگی ممکن است توسط کاربری در همان شبکه به کمک برنامه‌های ویژه‌ی استراق‌سمع بسته<sup>1</sup> دریافت شوند. راه‌حل عمومی، استفاده از سرویس‌ها و پروتکل‌های OpenSSH، SFTP و FTPS است. همچنین دستور زیر را اجرا کنید تا NIS، RSH و دیگر سرویس‌های منسوخ را پاک کنید:

```
# yum erase inetd xinetd ypsserv tftp-server telnet-server rsh-serve
```

▪ **کمینه‌سازی برنامه‌ها برای به کاستن از نفوذپذیری:** آیا واقعا به همه‌ی سرویس‌های وبی که روی سیستم نصب شده است، نیاز خواهید داشت؟ برای کاهش نفوذپذیری سیستم، از نصب نرم‌افزارهای اضافی خودداری کنید. از ابزار مدیریت بسته مانند yum، apt-get یا dpkg برای بررسی بسته‌های نصب شده روی سیستم به‌فرم زیر استفاده کنید:

<sup>1</sup> Packet Sniffer

```
# yum list installed
# yum list packageName
# yum remove packageName
```

و یا در نمونه‌های مبتنی بر توزیع Debian:

```
# dpkg --list
# dpkg --info packageName
# apt-get remove packageName
```

▪ **یک سرویس شبکه‌ای روی هر سیستم یا هر ماشین مجازی:** سرویس‌های شبکه‌ای مختلف را روی سرویس‌دهنده‌های مختلف یا ماشین‌های مجازی مختلف اجرا کنید. این کار تعداد سرویس‌هایی را که ممکن است آسیب ببینند، محدود می‌کند. در این حالت برای نمونه، اگر یک خرابکار وارد شبکه شود و بتواند در سرویس‌دهنده‌ی وب آپاچی نفوذ کند، قادر نخواهد بود تا در دیگر سرویس‌های شبکه مانند MySQL یا سرویس‌دهنده‌ی پست‌الکترونیکی نفوذ کند.

▪ **هسته و برنامه‌های سیستم را به‌روز نگه دارید:** اعمال وصله‌های امنیتی، مهم‌ترین بخش نگه‌داری یک سیستم لینوکسی است. لینوکس همه‌ی ابزارهای لازم برای به‌روزنگه‌داشتن سیستم را فراهم می‌کند و اجازه‌ی ارتقای آسان بین نسخه‌های مختلف را می‌دهد. همه‌ی به‌روزرسانی‌های امنیتی باید بررسی و در اسرع وقت اعمال شوند و در ادامه از ابزار مدیریت بسته مانند yum یا apt برای به‌روزنگه‌داشتن سیستم استفاده گردد:

```
# yum update
```

یا

```
# apt-get update && apt-get upgrade
```

می‌توانید توزیع CentOS، Red Hat و یا Fedora خود را طوری تنظیم کنید تا خبر انتشار یک به‌روزرسانی برای هر یک از بسته‌های yum را از طریق پست‌الکترونیکی به آگاهی شما رساند. گزینه‌ی دیگر، اعمال همه‌ی به‌روزرسانی‌ها از طریق وظایف ابزار cron است. برای سیستم‌های Debian و Ubuntu از ابزار apticron برای آگاه شدن از وجود به‌روزرسانی‌ها استفاده کنید.

▪ **از افزونه‌های امنیتی لینوکس استفاده کنید:** لینوکس از وصله‌های امنیتی مختلفی برای محافظت در قبال تنظیم‌های نادرست یا برنامه‌های مشکل‌ساز استفاده می‌کند. اگر ممکن است، از SELinux یا دیگر ابزارهای امنیتی اضافی لینوکس برای تشدید محدودیت‌های برنامه‌های شبکه و دیگر برنامه‌ها استفاده کنید. برای نمونه، SELinux از سیاست‌های امنیتی گوناگونی برای هسته‌ی لینوکس استفاده می‌کند. استفاده از SELinux که یک کنترل دسترسی اجباری (MAC)<sup>1</sup> را فراهم می‌کند، پیشنهاد می‌شود. در حالت استاندارد کنترل دسترسی احتیاطی (DAC)<sup>2</sup>، برنامه یا پردازش‌ای که به‌عنوان یک کاربر (UID یا SUID) اجرا می‌شود، تمام مجوزهای کاربر را برای دسترسی به اشیائی مانند فایل‌ها، سوکت‌ها و دیگر پردازش‌ها دارد. اجرای کنترل دسترسی اجباری، سیستم را از بدافزارهایی که ممکن است آن‌را تخریب یا نابود کنند، جلوگیری می‌کند. در فصل دوم کتاب به بررسی و تشریح کامل پیکربندی SELinux خواهیم پرداخت.

<sup>1</sup> Mandatory Access Control

<sup>2</sup> Discretionary Access Control

سیاست حساب‌های کاربری و رمزهای عبور قوی: از دستور `useradd` و `usermod` برای افزودن یک کاربر یا نگهداری یک حساب کاربری استفاده کنید. مطمئن شوید که از یک سیاست رمز عبور مناسب و قدرتمند استفاده می‌کنید. برای نمونه، یک گذرواژه‌ی مناسب دست‌کم شامل هشت حرف و ترکیبی از حروف الفبای کوچک و بزرگ، اعداد و کاراکترهای خاص است. از همه مهم‌تر گذرواژه‌های انتخاب کنید که بتوانید آن‌را به‌خاطر بسپارید. از ابزارهایی چون `John the ripper` برای پی‌بردن به گذرواژه‌های ضعیف کاربران روی سرور استفاده کنید. `pam_cracklib.so` را برای اجباری کردن یک سیاست گذرواژه‌ی قوی، پیکربندی کنید.

طول عمر رمزهای عبور: دستور `chage` تعداد روزهای بین تغییرات گذرواژه و تاریخی که آخرین بار گذرواژه تغییر کرده است را تغییر می‌دهد. این اطلاعات به‌وسیله‌ی سیستم استفاده می‌شود تا مشخص کند یک کاربر چه زمانی باید رمز عبورش را عوض کند. فایل `/etc/login.defs` پیکربندی رمزهای عبور مربوط به سایت، چون طول عمر آنها، را برعهده دارد. برای غیرفعال کردن تعیین طول عمر رمز عبور، دستور زیر را وارد کنید:

```
# chage -M 99999 userName
```

در آخر می‌توانید فایل `/etc/shadow` را طبق الگوی کلی زیر ویرایش نمایید:

```
: {password}: {lastpasswordchanged}: {Minimum_days}: {Maximum_days}: {Warn}: {Inactive}
: {expire}
```

گزینه‌های بالا با توجه به نام عبارت‌ها نیازی به توضیح ندارند، اما پیشنهاد می‌شود به‌جای ویرایش فایل `/etc/shadow` از دستور `chage` طبق الگوی کلی زیر استفاده کنید:

```
# chage -M 60 -m 7 -W 7 userName
```

محدود کردن استفاده از گذرواژه‌های پیشین: می‌توان کاربران را از استفاده‌ی دوباره از گذرواژه‌ی پیشین منع کرد. پارامتر `remember` در ماژول `pam_unix` می‌تواند برای پیکربندی تعداد گذرواژه‌های پیشین که نمی‌توانند دوباره استفاده شوند، مورد استفاده قرار گیرد.

قفل کردن یک کاربر پس از ورود ناموفق: در سیستم‌عامل لینوکس می‌توان از دستور `faillog` برای دیدن گزارش‌های ورود ناموفق یا اعمال محدودیت‌های ورود ناموفق استفاده کرد. `faillog` محتویات فایل ویژه‌ی ثبت وقایع و گزارش (`log`) را از گزارش‌های خطای ورود بانک اطلاعات `/var/log/faillog` به‌روز می‌کند. همچنین `faillog` برای نگهداری تعداد خطاهای ورود و اعمال محدودیت استفاده می‌شود. برای دیدن تلاش‌های ناموفق در ورود به سیستم، از دستور زیر استفاده نمایید:

```
# faillog
```

همچنین برای بازکردن یک حساب کاربری قفل شده پس از تلاش در ورود ناموفقش، از دستور زیر استفاده کنید:

```
# faillog -r -u userName
```

توجه داشته باشید که می‌توان از دستور `passwd` نیز برای قفل کردن و باز کردن یک حساب کاربری استفاده کرد. برای قفل کردن یک حساب کاربری به‌فرم زیر عمل می‌شود:

```
# passwd -l UserName
```

برای بازکردن یک حساب کاربری نیز از دستور زیر استفاده می‌شود:

```
# passwd -u UserName
```

➤ چگونه حساب‌های کاربری بدون رمز را پیدا کنیم؟ برای انجام این کار باید از دستور زیر استفاده نمایید:

```
# awk -F: '($2 == "") {print}' /etc/shadow
```

در ادامه با کمک دستور Passwd، حساب‌های بدون گذرواژه را قفل کنید.

➤ مطمئن شوید شناسه‌ی هیچ کاربری به‌جز کاربر ریشه، عدد صفر نباشد: تنها کاربر ریشه با شناسه کاربری صفر امکان دسترسی کامل به سیستم را دارد. برای پیدا کردن تمام حساب‌های کاربری با شناسه‌ی صفر دستور زیر را وارد کنید:

```
# awk -F: '($3 == "0") {print}' /etc/passwd
```

پس از اجرا باید تنها یک سطر مانند نمونه‌ی زیر را مشاهده کنید:

```
root:x:0:0:root:/root:/bin/bash
```

چنانچه حساب‌های دیگری از شناسه‌ی کاربری صفر استفاده می‌کنند، آنها را حذف کنید یا مطمئن شوید که آنها را می‌شناسید.

▪ **ورود کاربر ریشه را غیرفعال کنید:** هیچ وقت با کاربر ریشه وارد سیستم نشوید. باید از دستور sudo برای اجرای دستورهای در سطح کاربر ریشه استفاده کنید. دستور sudo به‌شدت امنیت سیستم را بالا می‌برد، به‌دلیل اینکه دیگر لازم نیست رمز کاربر ریشه را در اختیار دیگر مدیران سیستم قرار دهید. افزون بر این قابلیت پیگیری دستورهای استفاده شده را دارد.

▪ **امنیت فیزیکی سرور:** باید سرور را از نظر دسترسی فیزیکی به کنسول محافظت کنید. بایوس را برای غیرفعال کردن امکان راه‌اندازی سیستم از طریق CD، DVD، یا حافظه‌های USB تنظیم کنید. همچنین روی بایوس و Grub، گذرواژه‌ای را تنظیم کنید.

▪ **سرویس‌هایی را که نیاز ندارید، غیرفعال کنید:** سرویس‌ها و دایمون‌هایی (سرویس‌هایی که در پس‌زمینه اجرا می‌شوند) را که نیاز ندارید، غیرفعال کنید. همچنین باید همه‌ی این سرویس‌های ناخواسته را از بخش راه‌اندازی سیستم خارج نمایید. دستور زیر را برای فهمیدن اینکه چه دستورهایی در سطح سه اجرا<sup>1</sup> خواهند شد، اجرا کنید:

```
# chkconfig --list | grep '3:on'
```

برای غیرفعال کردن این سرویس‌ها می‌توان از دستورهای زیر استفاده کرد:

```
# service serviceName stop
# chkconfig serviceName off
```

همچنین پورت‌هایی که به شبکه گوش می‌دهند، را ببندید. از دستور زیر برای یافتن پورت‌های باز شبکه و برنامه‌های مربوط به آن استفاده کنید:

```
# netstat -tulpn
```

یا

```
# nmap -sT -O localhost
```

---

<sup>1</sup>\_Run Level 3

```
# nmap -sT -O server.example.com
```

از iptables برای بستن پورت‌های باز یا از دستورهای service و chkconfig برای توقف سرویس‌هایی که به این پورت‌ها گوش می‌کنند، استفاده کنید.

- **محیط X را پاک کنید:** به‌طور معمول در سرورها به محیط X نیازی نیست. دلیلی برای استفاده از محیط X در سرور اختصاصی پست‌الکترونیکی و یا سرور وب آپاچی وجود ندارد. می‌توان محیط X را غیرفعال یا حذف کرد تا امنیت و بازدهی سرور بالا رود. فایل /etc/inittab را ویرایش کنید و سطح پیش‌فرض را به سه تغییر دهید. سرانجام محیط X را با کمک دستور زیر حذف کنید:

```
# yum groupremove "X Window System"
```

- **پیکربندی iptables و TCPWrappers:** ابزار iptables برنامه‌ای است که در فضای کاربر User Space اجرا می‌شود و به کاربر اجازه‌ی پیکربندی دیوارآتش گنجانده شده در هسته‌ی لینوکس (Netfilter) را می‌دهد. از دیوارآتش برای پالایش ترافیک و صدور مجوز تنها برای ترافیک لازم، استفاده می‌شود. از TCPWrappers برای فیلتر کردن دسترسی شبکه به اینترنت استفاده کنید. از بسیاری از حمله‌های نوع DoS می‌توان به کمک ابزار iptables جلوگیری کرد.

- **مقاوم‌سازی /etc/sysctl.conf:** فایل /etc/sysctl.conf برای پیکربندی هسته‌ی لینوکس در زمان اجرا استفاده می‌شود. هسته، تنظیمات صورت گرفته در این فایل را هنگام راه‌اندازی سیستم می‌خواند و پیاده می‌کند. یک فایل نمونه به‌صورت زیر است:

```
#Turn on execshiel d
kernel . exec-shiel d=1
kernel . randomi ze_va_space=1
#Enabl e IP_spoofi ng protecti on
net. i pv4. conf. al l . rp_fi l lter=1
#Di sabl e IP_sourc e routi ng
net. i pv4. conf. al l . accept_sourc e_rout e=0
#I gnori ng broadc asts request
net. i pv4. i cmp_echo_i gnore_broadc asts=1
net. i pv4. i cmp_i gnore_bogus_err or_messag es=1
#Make sure spoofed packets get l ogged
net. i pv4. conf. al l . l og_marti ans =1
```

- **پارتیشن‌های جداگانه:** جداسازی سیستم‌های فایلی سیستم‌عامل از فایل‌های کاربران، می‌تواند سیستم بهتر و امن‌تری را برای کاربر به ارمغان بیاورد. مطمئن شوید که سیستم‌های فایلی /usr، /home، /var و /tmp در پارتیشن‌های جداگانه‌ای در اصطلاح Mount شده‌اند. پارتیشن‌های جداگانه برای مسیر ریشه‌ی آپاچی و سرویس‌دهنده‌ی FTP بسازید. فایل /etc/fstab را ویرایش کنید و مطمئن شوید که گزینه‌های پیکربندی زیر را در آن فایل پیاده کرده‌اید:

➤ noexec: اجازه‌ندادن به اجرای فایل‌های دودویی در این پارتیشن برای جلوگیری از اجرا شدن فایل‌های موجود در آن پارتیشن به‌جز فایل‌های اسکریپت.

➤ nodev: اجازه‌ندادن به کاراکتر یا دستگاه‌های ویژه در این پارتیشن برای پیش‌گیری از استفاده از فایل دستگاه‌هایی چون sda, zero و غیره.

➤ nosuid: اجازه‌ندادن به دسترسی به SUID و SGID در این پارتیشن برای جلوگیری از فعال شدن بیت .setuid



یک مثال از فایل `/etc/fstab` برای محدود کردن دسترسی کاربری به پارتیشن `/dev/sda5` که پارتیشن ریشه‌ی سرویس‌دهنده‌ی FTP است، به فرم زیر می‌باشد:

```
/dev/sda5 /ftpdata ext3 default, nosuid, nodev, noexec 1 2
```

همچنین عملیات مربوط به سهمیه‌بندی دیسک را به‌دقت انجام دهید. مطمئن شوید که سهمیه‌بندی دیسک برای تمام کاربران فعال شده باشد. برای پیاده‌سازی سهمیه‌بندی دیسک گام‌های زیر را دنبال کنید:

- سهمیه‌بندی برای هر سیستم‌فایل را با تغییر `/etc/fstab` اعمال کنید.
  - سیستم‌های فایل را دوباره Mount نمایید.
  - پایگاه‌داده سهمیه‌ها را ایجاد کنید و جدول استفاده از دیسک را بسازید.
  - سیاست‌های سهمیه‌بندی را پیاده نمایید.
- **قابلیت IPv6 را غیرفعال کنید:** IP نسخه شش، یک لایه اینترنتی جدید از پروتکل TCP/IP است که جایگزین IP نسخه چهار خواهد شد و مزایای زیادی نیز به‌همراه خواهد داشت. هم‌اکنون ابزارهای مناسبی برای بررسی مشکلات امنیتی یک سیستم IPv6 موجود نیست. شمار زیادی از توزیع‌ها اقدام به فعال کردن IPv6 به‌طور پیش‌فرض کرده‌اند. به‌دلیل اینکه بیشتر مدیران سیستم IP نسخه شش را نظارت نمی‌کنند، نفوذگران می‌توانند یک ترافیک بد را از طریق IPv6 وارد شبکه کنند. بنابراین، یا IPv6 را غیرفعال کنید و یا دیوارآتش IPv6 لینوکس را پیکربندی نمایید.
  - **فایل‌های SUID و SGID ناخواسته را غیرفعال کنید:** تمام فایل‌هایی که بیت SUID/SGID آنها فعال شده، می‌توانند در صورت وجود مشکل امنیتی، مورد سوءاستفاده قرار گیرند. همه‌ی کاربران محلی یا راه‌دور می‌توانند از این فایل‌ها استفاده کنند. ایده‌ی خوبی است که همه‌ی این فایل‌ها یافت شود. از دستور `find` برای این منظور استفاده کنید:

یافتن همه‌ی فایل‌هایی که بیت `user id` آنها فعال شده:

```
# find / -perm +4000
```

پیداکردن تمام فایل‌هایی که بیت `group id` آنها فعال شده:

```
# find / -perm +2000
```

و یا برای پیداکردن هر دوی آنها:

```
# find / \( -perm -4000 -o -perm -2000 \) -print
```

```
# find / -path -prune -o -type f -perm +6000 -ls
```

باید در مورد هر فایل گزارش شده، تحقیق شود. برای کسب اطلاعات بیشتر راهنمای آن فایل‌ها را بخوانید.

- فایل‌های قابل نوشتن عمومی: هر فایل قابل نوشتن عمومی می‌تواند توسط هر کسی مورد استفاده قرار گیرد و این باعث به‌وجود آمدن مشکلات امنیتی خواهد شد. با کمک دستور زیر فایل‌های قابل نوشتن عمومی و با بیت‌های `sticky` فعال شده را بیابید:

```
# find /dir -xdev -type d \( -perm -0002 -a ! -perm -1000 \) -print
```

نیاز است که درباره‌ی هر فایل گزارش شده تحقیق کنید و سپس بیت SUID و GUID آنها را تنظیم یا پاک کنید.

➤ فایل‌های بدون مالک: فایل‌های بدون مالک می‌توانند مشکل امنیتی به‌وجود بیاورند. با کمک دستور زیر، این فایل‌ها را که به هیچ کاربر یا گروه معتبری تعلق ندارند، پیدا کنید:

```
# find /dir -xdev \( -nouser -o -nogroup \) -print
```

پس از آن باید در مورد گزارش‌های خروجی این دستور تحقیق کنید و مالکیت فایل‌ها را به کاربران ویا گروه‌های شناخته شده انتقال دهید.

▪ **از یک سرویس تأیید هویت مرکزی استفاده کنید:** بدون یک سیستم هویت‌سنجی متمرکز، داده‌های هویت‌سنجی کاربران، ناپایدار و غیرقابل اعتماد می‌شوند. این امر باعث به‌وجود آمدن امکان دسترسی برای حساب‌های تاریخ گذشته یا حساب‌های فراموش شده‌ای می‌شود که باید از روی سیستم پاک می‌شدند. یک سرویس تأیید هویت مرکزی به کاربر اجازه می‌دهد تا امکان کنترل و نگاه‌داری حساب‌های یونیکس و لینوکس و داده‌های هویت‌سنجی آنها را به‌صورت متمرکز در اختیار داشته باشد. همچنین می‌توان داده‌ها را بین سرورهای خود همسان<sup>1</sup> کرد. از سرویس NIS برای این سیستم تأیید هویت مرکزی استفاده نکنید و به‌جای آن OpenLDAP را برای سرورها و کاربران به کار ببرید. Kerberos، هویت‌سنجی را به‌صورت یک سرویس طرف سوم قابل اعتماد و با استفاده از رمزگذاری و با این فرض انجام می‌دهد که بسته‌های شبکه در فضایی ناامن منتقل می‌شوند و ممکن است توسط هر کسی خوانده، ویرایش و دستکاری شوند. Kerberos، براساس رمزنگاری با کلید متقارن کار می‌کند و به یک مرکز توزیع کلید نیاز دارد. به کمک این ابزار می‌توان از راه دور وارد سیستم شد، فایل‌ها را کپی کرد و کپی فایل درون سیستمی و دیگر عملیات با خطر بالا را امن‌تر نمود. به این ترتیب زمانی که کاربرانی با استفاده از Kerberos وارد یکی از سرویس‌های شبکه می‌شوند، کسانی که سعی می‌کنند با پایش اطلاعات شبکه گذرواژه‌ها را بدزدند، ناکام می‌مانند.

▪ **گزارش‌گیری و حسابرسی:** باید سیستم گزارش‌گیری و حسابرسی را برای جمع‌آوری اطلاعات مربوط به تلاش‌های نفوذ تنظیم کنید. به‌صورت پیش‌فرض گزارش‌های سیستمی در مسیر /var/log ذخیره می‌شوند. این کار همچنین برای پی‌بردن به پیکربندی نامناسب نرم‌افزار که سیستم را برای حمله آماده می‌سازد، مفید است. خواندن راهنماهای مربوط به دایمون syslogd و فایل پیکربندی syslog.conf نیز در این باره مفید خواهد بود.

➤ گزارش‌های مشکوک را با کمک ابزارهای logwatch و logcheck بررسی کنید. این ابزارها بررسی گزارش‌ها را ساده‌تر می‌سازد و درباره‌ی موارد نامعمول، توضیحات جزئی‌تری فراهم می‌نماید. نمونه‌ای از سبک استفاده و خروجی ابزار logwatch در قالب شکل (1-1) نمایش داده شده است.

➤ حسابرسی سیستم با استفاده از auditd: دایمون auditd برای انجام حسابرسی سیستم ایجاد شده است. auditd مسئول نوشتن گزارش‌های audit بر روی دیسک است. در حین فرآیند راه‌اندازی سیستم قواعد موجود در /etc/audit.rules توسط سرویس پس‌زمینه‌ی auditd خوانده می‌شود. می‌توان فایل /etc/audit.rules را باز و تغییرات مورد نظر مانند تنظیم محل فایل گزارش و دیگر گزینه‌ها را تنظیم نمود. توسط دایمون auditd می‌توان از موارد زیر آگاه شد:

○ رخدادهای هنگام راه‌اندازی و خاموش شدن سیستم

<sup>1</sup>\_Sync

- تاریخ و زمان یک رخداد
- نام کاربری که مسئول یک رخداد است
- نوع رخداد (ویرایش، دسترسی، پاک کردن، نوشتن و غیره)
- موفقیت یا شکست یک رخداد
- ثبت رخدادهایی که تاریخ و زمان را تغییر می‌دهند.
- آگاهی از اینکه چه کسی تنظیمات شبکه سیستم را تغییر داده است.
- ثبت رخدادهایی که اطلاعات کاربر/گروه را تغییر می‌دهند.
- آگاهی از اینکه چه کسی یک فایل را تغییر داده است.

```
# logwatch --service sshd --range=Today --detail=High

----- SSHD Begin -----

Illegal users from:
192.168.1.83: 12 times
bob/password: 6 times
george/password: 3 times
raphael/password: 3 times

**Unmatched Entries**
pam_succeed_if(sshd:auth): error retrieving information about user raphael :
3time(s)
pam_succeed_if(sshd:auth): error retrieving information about user bob : 6
time(s)
PAM 2 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser=
rhost=192.168.1.83 : 4 time(s)
pam_succeed_if(sshd:auth): error retrieving information about user george : 3
time(s)

----- SSHD End -----
```

شکل (1-1) شمایی از نحوه‌ی استفاده و خروجی ابزار logwatch

- سرور OpenSSH خود را امن کنید: استفاده از پروتکل SSH برای ورود و انتقال فایل‌ها از راه دور پیشنهاد می‌شود. با توجه به اینکه SSH در معرض انواع حمله‌ها قرار دارد، تلاش کنید آن را تا حد امکان امن کنید. پروتکل SSH و سرور OpenSSH و همچنین مکانیزم‌های امن‌سازی آنها در فصل دوازدهم کتاب به‌طور کامل بررسی خواهد شد.
- سیستم‌های تشخیص نفوذ را نصب کرده و به‌کار ببرید: سیستم تشخیص نفوذ شبکه (NIDS)<sup>1</sup> سیستمی برای کشف فعالیت‌های مخرب چون حمله‌های DoS، پویش پورت‌ها و هر تلاشی برای شکستن سیستم، از طریق کنترل و نظارت ترافیک شبکه است. بهتر است تمام نرم‌افزارهای کنترل سلامت و یکپارچگی سیستم را پیش از فعال شدن سیستم در محیط کاری نصب و آزمایش کنید. حتی پیشنهاد می‌شود نرم‌افزار AIDE را پیش از اتصال سرور به هر شبکه‌ای نصب کنید. IDE یک سیستم تشخیص نفوذ مبتنی بر میزبان است. این برنامه می‌تواند سازوکار درونی سیستم را پایش کند. Snort نیز نرم‌افزاری است که می‌تواند تحلیل بی‌درنگ بسته‌ها در شبکه‌ی IP را برای کشف نفوذ، به انجام برساند. فصل دهم کتاب به بررسی سیستم‌های تشخیص نفوذ، HoneyPot و مانیتورینگ شبکه خواهیم پرداخت.

<sup>1</sup> Network Intrusion Detection System

- **محافظت از پوشه‌ها، فایل‌ها و نامه‌های الکترونیکی:** لینوکس محافظت‌های عالی در برابر دستیابی به اطلاعات برای افراد احرازهویت نشده دارد و با کمک سطوح دسترسی فایل‌ها و MAC از دستیابی افراد ناشناس به فایل‌ها جلوگیری می‌کند؛ هر چند مجوزهای لینوکس، چنانچه حمله‌کننده به سیستم دسترسی فیزیکی داشته باشد و بتواند دیسک سخت آن‌را به کامپیوتر دیگری منتقل کند، هیچ سودی نخواهد داشت، اما با ابزارهای دیگری می‌توان ضریب امنیت را بالا برد. این ابزارها شامل موارد زیر می‌باشند:
  - استفاده از gpg برای رمزنگاری و رمزگشایی فایل‌ها به‌وسیله‌ی گذرواژه
  - گذرواژه‌های لینوکسی و یونیکسی از فایل‌ها در هنگام استفاده از OpenSSL یا دیگر ابزارهای مشابه محافظت می‌کنند.
  - encryptfs می‌تواند برای رمزنگاری پوشه‌ها استفاده شود.
  - True Crypt یک برنامه‌ی رمزنگاری دیسک کدباز برای سیستم‌های لینوکسی، ویندوزی و MacOS است. می‌توان از گواهی‌های SSL و کلیدهای gpg برای امن‌سازی ارتباطات پست‌الکترونیکی بین هر دو طرف سرور و کلاینت استفاده کرد.

## 6-1 مهم‌ترین نقاط آسیب‌پذیر یونیکس و لینوکس

در این بخش به‌طور کلی به معرفی مهم‌ترین نقاط آسیب‌پذیر یونیکس و لینوکس خواهیم پرداخت. در این راستا، در فصل‌های بعدی کتاب به بررسی کامل هریک از نقاط آسیب‌پذیر اشاره شده، علت وجود ضعف امنیتی، سیستم‌های عامل در معرض تهدید، روش‌های تشخیص آسیب‌پذیری سیستم و سبک مقابله و یا پیشگیری در مقابل هریک از نقاط آسیب‌پذیر می‌پردازیم. همان‌گونه که بخش‌های پیشین به آن اشاره شد، بیشتر تهدیدها و حمله‌ها، متأثر از وجود نقاط آسیب‌پذیر در سیستم‌های عامل می‌باشد که زمینه‌ی تهاجم را برای مهاجمان فراهم می‌آورد. شناسایی و تجزیه و تحلیل نقاط آسیب‌پذیر در هریک از سیستم‌های عامل، نتیجه‌ی تلاش و پردازش ده‌ها کارشناس امنیتی ورزیده در سطح جهان است و باید مدیران سیستم و شبکه در یک سازمان به‌سرعت با آنها آشنا شوند و اقدام‌های لازم را انجام دهند. نقاط آسیب‌پذیر موجود در هر سیستم‌عامل مبتنی بر یونیکس و لینوکس که در ادامه به آنها اشاره می‌گردد، سندی پویا و شامل دستورالعمل‌های لازم برای برخورد مناسب با هریک از نقاط آسیب‌پذیر می‌باشد. همان‌گونه که می‌دانید یونیکس و لینوکس، سیستم‌عامل‌هایی رایج در جهان هستند که امروز در سطح بسیار گسترده‌ای استفاده می‌شوند. تاکنون حمله‌های فراوانی توسط مهاجمان متوجه این نوع سیستم‌های عامل بوده است. با توجه به حمله‌های متنوع و گسترده‌ی انجام شده، می‌توان مهم‌ترین نقاط آسیب‌پذیر یونیکس و لینوکس را به ده گروه عمده‌ی زیر تقسیم نمود:

- BIND Domain Name System
- Remote Procedure Calls (RPC)
- Apache Web Server
- General UNIX Authentication Accounts with No Passwords or Weak Passwords
- Clear Text Services
- Sendmail
- Simple Network Management Protocol (SNMP)
- Secure Shell (SSH)
- Misconfiguration of Enterprise Services NIS/NFS
- Open Secure Sockets Layer (SSL)

## 7-1 امنیت سیستم‌عامل یونیکس

یونیکس به‌عنوان وارث ساده و کارآتر مالتیکس، ویژگی‌های امنیتی فراوانی مانند گذرواژه‌های قوی، استفاده از حلقه‌های حفاظت، لیست کنترل دسترسی و غیره را از مالتیکس اقتباس کرد، اما این ویژگی‌ها ساده‌تر و کارآتر بودند. به‌دلیل اینکه یونیکس مانند سیستم‌عامل مالتیکس پروژه‌ای دولتی نبود، با اهداف امنیتی متفاوتی طراحی شد. هدف یونیکس، ایجاد سکوی مشترکی (برای نمونه، دستگاه‌ها و سیستم‌فایل) بود که می‌توانست توسط کاربران مختلف به اشتراک گذاشته شود. در نتیجه، مسئله‌ی امنیت به محافظت تبدیل شد که در آن هدف، محافظت از داده‌های کاربران در برابر خطاهای غیرعمدی در برنامه‌هایشان می‌باشد. هرچند، محافظت تضمین نمی‌کند که می‌توان به اهداف محرمانگی و جامعیت (یعنی امنیت) دست یافت. اجرای اهداف امنیتی مستلزم این است که مکانیزم‌های امنیتی سیستم بتوانند اهداف امنیتی را حتی هنگامی که تمام نرم‌افزارهای بیرون از پایگاه محاسبه، معتمدی مغرض باشند، اجرا کنند. در نتیجه، وقتی سیستم‌های یونیکس از طریق اینترنت به کاربران نامعتبر متصل شدند، بسیاری از تصمیم‌های طراحی که برای محافظت اتخاذ شده بودند، کاربرد چندانی نداشتند. مکانیزم‌های امنیتی یونیکس، قابلیت برآورده ساختن نیازمندی‌های یک سیستم‌عامل امن را ندارند. برای گسترش یا جایگزینی مکانیزم‌های ناامن در سیستم‌های یونیکس متداول با مکانیزم‌هایی که امکان برآورده ساختن نیازمندی‌های یک سیستم‌عامل امن را دارند، تلاش‌های زیادی انجام شده است. در ادامه به بررسی برخی از این مکانیزم‌ها درباره‌ی امنیت یونیکس می‌پردازیم.

### 7-1-1 امنیت یونیکس

پیش از بررسی جزئیات امنیتی یونیکس، توصیف کوتاهی از سیستم یونیکس ارائه می‌کنیم. یک سیستم یونیکس در حال اجرا، از یک هسته‌ی سیستم‌عامل و شمار زیادی فرآیند که هر یک از آنها برنامه‌ای را اجرا می‌کنند، تشکیل شده است. یک رمز حلقه‌ی حفاظت، هسته‌ی یونیکس را از فرآیندها ایزوله می‌کند. هر فرآیند، فضای آدرس ویژه‌ی خود را دارد که آدرس‌هایی از حافظه را که آن فرآیند می‌تواند به آنها دسترسی داشته باشد، مشخص می‌کند. سیستم‌های یونیکس نوین، اصولاً فضاهای آدرس را برحسب مجموعه صفحات حافظه که فرآیندها می‌توانند به آنها دسترسی داشته باشند، تعریف می‌کنند. یونیکس از مفهوم فایل برای تمام اشیای پایای سیستم مانند دستگاه‌های I/O، شبکه و ارتباطات بین فرآیندی استفاده می‌کند. به هر فرآیند یونیکس، بر اساس کاربر مربوطه، یک شناسه تخصیص داده می‌شود و دسترسی به فایل‌ها با استفاده از شناسه‌ی فرآیند، محدود می‌شود.

در امنیت یونیکس، هدف، محافظت کاربران از یکدیگر و محافظت برپایه‌ی محاسبه‌ی معتبر (TCB)<sup>1</sup> از تمام کاربران می‌باشد. به بیانی غیر رسمی، TCB یونیکس، از هسته و شمار زیادی فرآیند که با شناسه‌ی کاربر ممتاز (root) یا کاربر مافوق (اجرا می‌شوند، تشکیل شده است. فرآیندهای root، سرویس‌های مختلفی مانند راه‌اندازی سیستم، احراز هویت کاربر، مدیریت، سرویس‌های شبکه و غیره را فراهم می‌سازد. هم هسته و هم فرآیندهای root، مجوز دسترسی کامل به سیستم را دارند. تمام فرآیندهای دیگر، بر اساس شناسه‌ی کاربر مربوطه، دارای دسترسی محدودی می‌باشند.

### 7-2-1 سیستم حفاظت یونیکس

یونیکس یک سیستم حفاظت کلاسیک و نه یک سیستم حفاظت امن پیاده‌سازی می‌کند. سیستم حفاظت یونیکس از یک وضعیت حفاظت و مجموعه‌ای از عملیات که امکان تغییر وضعیت را فراهم می‌سازند، تشکیل شده است. در نتیجه، یونیکس یک سیستم کنترل دسترسی اختیاری (DAC) است. با این وجود، یونیکس برخی از جنبه‌های سیستم حفاظت

<sup>1</sup> Trusted Computing Base

امن را دارا می‌باشد. نخست اینکه، سیستم حفاظت یونیکس یک وضعیت گذار تعریف می‌کند که مشخص می‌سازد که فرآیندها چگونه بین حوزه‌های حفاظت تغییر می‌کنند. دوم اینکه، وضعیت برچسب زنی معمولاً برای موارد خاص استفاده می‌شود. سرویس‌های معتبر، شناسه‌های کاربری را به فرآیندها تخصیص می‌دهند، اما کاربران می‌توانند تخصیص مجوزها به منابع سیستم (یعنی فایل‌ها) را کنترل کنند. به‌عنوان تحلیل پایانی، این مکانیزم‌ها و سیستم حفاظت اختیاری، برای ایجاد سیستمی که نیازمندی‌های یک سیستم‌عامل امن را برآورده سازد، ناکافی هستند. وضعیت حفاظت، عملیاتی را که نهادهای سیستم می‌توانند روی اشیای سیستم انجام دهند، مشخص می‌کند. وضعیت حفاظت یونیکس، دسترسی نهادهای فایل‌ها (اشیا) را به شناسه‌های فرآیند (نهادهای) تخصیص می‌دهد. هر شناسه‌ی فرآیند یونیکس، از یک شناسه‌ی کاربر (UID) یک شناسه‌ی گروه (GID) و یک مجموعه گروه‌های پیوستی، تشکیل شده است. از ترکیب این موارد برای تعیین حق دسترسی استفاده می‌شود.

همه‌ی منابع یونیکس، به‌عنوان فایل نشان داده می‌شوند. وضعیت حفاظت، نهادهایی را مشخص می‌کند که می‌توانند عملیات خواندن، نوشتن و اجرا کردن را (با معنای استاندارد این عملیات) روی فایل‌ها انجام دهند. با اینکه دایرکتوری‌ها فایل نیستند، اما به‌عنوان فایل نشان داده می‌شوند، هرچند عملیات، معناهای متفاوتی دارند (مثلاً برای یک دایرکتوری، عمل اجرا به معنی جست‌وجو است). همچنین به فایل‌ها یک UID و یک GID نسبت داده می‌شود که امتیازهای خاص فرآیندهایی با این شناسه‌ها را نشان می‌دهند. یک فرآیند با UID مالک، می‌تواند هر بُعدی از وضعیت حفاظت را برای این فایل تغییر دهد. فرآیندهایی با هر یک از شناسه‌های UID مالک یا GID گروه، می‌توانند حقوق اضافه‌تری در دسترسی به فایل داشته باشند که در ادامه به تشریح آن می‌پردازیم. مجموعه‌ی محدود اشیا و عملیات، طراحان یونیکس را قادر می‌ساخت تا از یک لیست کنترل دسترسی فشرده به نام بیت‌های حالت یونیکس برای مشخص کردن حقوق دسترسی شناسه‌ها نسبت به فایل‌ها، استفاده کنند. بیت‌های حالت، حقوق سه نوع نهاد را مشخص می‌کردند که شامل موارد زیر می‌باشند:

1. UID مالک فایل

2. GID گروه فایل

3. دیگر نهادها

با استفاده از بیت‌های حالت، مجوزدهی به‌طریق زیر انجام می‌شود:

نخست، مکانیزم مجوزدهی یونیکس بررسی می‌کند که آیا UID شناسه‌ی فرآیند، برابر UID مالک می‌باشد؟ اگر چنین بود، از بیت‌های حالت مالک برای صدور مجوز دسترسی استفاده می‌کند. اگر GID شناسه‌ی فرآیند یا گروه‌های پیوستی، برابر GID گروه فایل بودند، آن‌گاه از بیت‌های حالت مجوزهای گروه، استفاده می‌شود؛ وگرنه، از مجوزهای تخصیص داده شده به دیگران استفاده می‌شود. همان‌گونه که می‌دانید، بیت‌های حالت یونیکس به شکل {بیت‌های دیگران، بیت‌های گروه، بیت‌های مالک} می‌باشند که هر عنصر این چندتایی، از یک بیت خواندن، یک بیت نوشتن و یک بیت اجرا تشکیل شده است. همان‌گونه که به آن اشاره شد، سیستم حفاظت یونیکس، یک سیستم کنترل دسترسی اختیاری است. به‌ویژه، این بدین معناست که بیت‌های حالت یک UID مالک، یا GID گروه می‌توانند با هر فرآیندی که توسط مالک فایل اجرا می‌شود (یعنی، فرآیندی که UID آن با مالک فایل یکسان است) تغییر داده شوند. اگر اعتماد داشته باشیم که تمام فرآیندهای کاربران، هر کاری که انجام می‌دهند به سود کاربر است، آن‌گاه اهداف امنیتی کاربر می‌توانند اجرا شوند. هرچند، این فرض یک فرض منطقی نیست.

امروز، کاربران انواع مختلفی از فرآیندها را اجرا می‌کنند که برخی از آنها ممکن است متعلق به یک حمله‌کننده باشد یا نسبت به به‌مخاطره افتادن توسط حمله‌کننده‌ها آسیب‌پذیر باشد؛ پس کاربر نمی‌تواند هیچ تضمینی داشته باشد که این فرآیندها با اهداف امنیتی تناقض نخواهند داشت. در نتیجه، یک سیستم‌عامل امن، برای اجرای اهداف امنیتی کاربر، نمی‌تواند از کنترل دسترسی اختیاری استفاده کند. چراکه کنترل دسترسی اختیاری، به کاربران اجازه می‌دهد که افزون بر بیت‌های حالت UID مالک، GID گروه مربوط به فایل‌هایشان را نیز تغییر دهند. در این حالت برچسب‌زنی فایل‌ها نیز اختیاری می‌باشد.

در یک سیستم حفاظت امن، وضعیت برچسب‌زنی باید اجباری باشد؛ پس این دلیل دیگری برای این است که سیستم یونیکس نیازمندی‌های امنیتی یک سیستم‌عامل امن را برآورده نمی‌سازد. فرآیندهای یونیکس، با استفاده از سرویس‌های معتمد با برچسب‌هایی از مجموعه‌ای از برچسب‌ها (یعنی UIDها و GIDها) که توسط مدیران معتمد تعریف شده‌اند، برچسب‌زنی می‌شوند و فرآیندهای فرزند، شناسه‌ی فرآیندهایشان را از والد خود به ارث می‌برند. چنین رهیافت اجباری در برچسب‌زنی فرآیندها با شناسه‌ها، نیازمندی‌های امنیتی یک سیستم‌عامل امن را برآورده می‌کند (گرچه، این رهیافت نسبتاً نامنعطف است). همان‌گونه که می‌دانید، بیت‌های حالت در یونیکس، خصوصیتی را نیز برای گذارهای حوزه‌ی حفاظت دارد که بیت setuid نامیده می‌شود. زمانی که این بیت برای یک فایل یک می‌شود، هر فرآیندی که فایل را اجرا کند، به‌طور خودکار یک گذار حوزه‌ی حفاظت به UID مالک فایل و GID گروه انجام خواهد داد. برای نمونه، اگر یک فرآیند root بیت setuid را روی فایل‌ای که آن فرآیند مالک آن می‌باشد یک کند، آنگاه هر فرآیندی که آن فایل را اجرا کند، با UID ریشه اجرا خواهد شد. به‌دلیل اینکه بیت UID یک بیت حالت است، مالک فایل می‌تواند آن را یک کند، بنابراین به روشی اختیاری مدیریت می‌شود. یک وضعیت حفاظت امن، نیاز به یک وضعیت گذار اجباری دارد که تمام گذارهای حوزه حفاظت را مشخص می‌کند، پس استفاده از بیت‌های اختیاری setuid ناکافی می‌باشد.

### 3-7-1 تحلیل امنیتی یونیکس

در صورتی یک سیستم‌عامل امن است که بتواند نیازمندی‌های امنیتی سیستم‌عامل امن را برآورده کند. یونیکس قادر به برآورده کردن هیچ‌یک از این نیازمندی‌ها که شامل موارد زیر می‌باشند، نیست:

- وساطت کامل: واسط ناظر مرجع چگونه تضمین می‌کند که تمام عملیات حساس امنیتی به‌درستی وساطت می‌شوند؟ واسط ناظر مرجع یونیکس از قلاب‌ها تشکیل شده است که مجوزهای دسترسی به فایل یا inode را در برخی فراخوان‌های سیستمی بررسی می‌کنند. واسط ناظر مرجع یونیکس، دسترسی به اشیایی را که هسته از آنها در عملیاتش استفاده می‌کند، مجوزدهی می‌نماید. مشکلی که وجود دارد این است که عملیات یونیکس (خواندن، نوشتن و اجرا) برای بیان کنترل دسترسی به اطلاعات، به اندازه‌ی کافی بامعنی و رسا نیستند. در یونیکس می‌توان بدون نیاز به داشتن مجوز نوشتن، فایل‌ها را تغییر داد (برای نمونه، با استفاده از `fcntl`).
- وساطت کامل: آیا واسط ناظر مرجع، تمام عملیات حساس امنیتی را روی تمام منابع سیستم وساطت می‌کند؟ مجوزدهی در یونیکس، وساطت کامل تمام منابع سیستم را فراهم نمی‌کند. برای برخی اشیاء مانند ارتباطات شبکه‌ای، خود یونیکس هیچ مجوزدهی را فراهم نمی‌سازد.
- وساطت کامل: چگونه اثبات می‌کنیم که واسط ناظر مرجع، وساطت کامل را فراهم می‌کند؟ به‌دلیل اینکه واسط ناظر مرجع در جایی قرار گرفته است که در آنجا تمام عملیات حساس امنیتی انجام می‌شوند، دانستن اینکه تمام

عملیات شناسایی شده‌اند و تمام مسیرها وساطت شده‌اند، کار مشکلی است. از هیچ رهیافتی برای اثبات وساطت کامل استفاده نشده است.

مقاوم بودن در برابر دستکاری: سیستم چگونه از ناظر مرجع خود (شامل سیستم حفاظتش) محافظت می‌کند؟ ناظر مرجع و سیستم حفاظتش در هسته نگه‌داری می‌شوند، اما این کار، مقاوم بودن در برابر دستکاری را تضمین نمی‌کند. نخست اینکه سیستم حفاظت اختیاری است، بنابراین هر فرآیند در حال اجرا می‌تواند آن را دست‌کاری کند. فرآیندهای غیرقابل اعتماد کاربر می‌توانند مجوزهای داده‌های کاربران خود را به‌طور دلخواه تغییر دهند، بنابراین اجرای اهداف امنیتی روی داده‌های کاربران امکان‌پذیر نیست. دوم اینکه هسته‌ی یونیکس به شیوه‌ای که هسته‌ی مالتیکس محافظت می‌شد، در برابر فرآیندهای غیرقابل اعتماد محافظت نمی‌شود. هر دو از حلقه‌های حفاظت برای ایزوله‌سازی استفاده می‌کنند، اما سیستم مالتیکس آشکارا دروازه‌هایی را مشخص می‌کند که برای بررسی قانونی بودن آرگومان‌های گذارهای حلقه استفاده می‌شوند. گرچه هسته‌های یونیکس بیشتر رویه‌هایی را برای بررسی صحت آرگومان‌های فراخوان‌های سیستمی فراهم می‌سازند، اما چنین رویه‌هایی احتمالاً در مکان‌های اشتباهی قرار گرفته‌اند. در آخر، فرآیندهای سطح کاربر از واسطه‌های مختلفی، بالاتر و فراتر از فراخوان‌های سیستمی، برای دسترسی و تغییر خود هسته استفاده می‌کنند. گستره‌ی این واسطه‌ها از توانایی در نصب پیمان‌های هسته در سیستم‌های فایل (مانند `/proc` یا `sysfs`) تا واسطه‌هایی میان سوکت‌های `netlink` برای دسترسی مستقیم به حافظه‌ی هسته (مانند از طریق دستگاه فایل `/dev/kmem`) می‌باشد. تضمین اینکه تنها کدهای معتمد می‌توانند به این واسطه‌ها دسترسی داشته باشند، ناممکن است.

مقاوم بودن در برابر دست‌کاری: آیا مکانیزم حفاظت سیستم از برنامه‌های نوع TCB محافظت می‌کند؟ یونیکس افزون بر هسته، تمام فرآیندهای `root` (شامل تمام فرآیندهایی که هنگام وارد شدن کاربر به‌عنوان کاربر `root` اجرا می‌شوند) را شامل می‌شود. به دلیل اینکه این فرآیندها می‌توانند هر نوع برنامه‌ای را اجرا کنند، محافظت TCB در برابر دست‌کاری را نمی‌توان تضمین کرد. حتی بدون در نظر گرفتن کاربران `root`، حجم کد TCB بسیار زیاد است و با تهدیدهای بسیاری مواجه است، از اینرو نمی‌توان ادعا کرد که TCB در برابر دست‌کاری مقاوم است. برای نمونه، فرآیندهای `root` فراوانی وجود دارند که پورت‌های شبکه بازی دارند که راه‌هایی برای به مخاطره انداختن این فرآیندها می‌باشند. اگر هر کدام از این فرآیندها به مخاطره بیافتند، سیستم یونیکس به مخاطره می‌افتد، زیرا هیچ محافظتی بین فرآیندهای `root` وجود ندارد. چنانکه در ادامه بیان می‌شود، به فرآیندهای `root` یونیکس نمی‌توان به اندازه‌ی کافی اعتماد داشت و محافظت شده نیستند، بنابراین به‌طور کلی امکان تغییر غیرمجاز سیستم حفاظت وجود دارد. در نتیجه، نمی‌توانیم در یونیکس یک سیستم حفاظت مقاوم در برابر دست‌کاری داشته باشیم.

قابل اثبات بودن: چه چیزی مبنای درستی TCB سیستم است؟ هر مبنایی برای اثبات درستی سیستم یونیکس، غیررسمی است. اندازه‌ی مرزبندی نشده‌ی TCB، مانع از هرگونه اثبات رسمی می‌شود. افزون بر این، اندازه و ماهیت گسترش‌پذیری هسته (مثلاً از طریق راه‌اندازهای دستگاه‌ها و دیگر ماژول‌های هسته‌ای جدید)، اثبات درستی هسته را ناممکن می‌سازد.

قابل اثبات بودن: آیا سیستم حفاظت، اهداف امنیتی را اجرا می‌کند؟ به دلیل عدم وساطت کامل و مقاوم نبودن در برابر دست‌کاری، نمی‌توان اجرای اهداف امنیتی را اثبات کرد. به دلیل اینکه نمی‌توانیم سیاستی به اندازه‌ی کافی غنی تعریف کنیم که از افشا یا تغییر غیرمجاز اطلاعات جلوگیری کند، نمی‌توانیم اهداف امنیتی محرمانگی و