مروری بر یونیکس:

برای آنکه بتوانید مکانیزم بسیاری از حملات را تجزیه و تحلیل کنید حداقل باید یک آشنایی کامل وبنیادی با سیستم عامل یونیکس داشته باشید.استفاده از این سیستم عامل بعنوان بستری برای سیستمها و سرویس دهنده های شبکه بسیار رایج و معمول است. به دلیل سابقه زیادی که دارد حملات متنوع و پیچیده ای بر علیه آن شکل گرفته است

یونیکس سیستم عاملی بسیار قدرتمند و زیباست بگونه ای که بسیاری از سیستم های عامل دیگر از اصول آن تقلید کرده اند!!!یونیکس حدود سی سال قبل در شرکت AT&Tطراحی شد و درطی روند تکمیلی سی ساله به سیستم عاملی تبدیل شد که میتواند داعیه ی قویترین و مطمئن ترین سیستم عامل جهان را داشته باشد بگونه ای که بسیاری از سرویس دهنده های حساس وعظیم دنیا از این سیسم عامل بهره گرفته اند. میتوان گفت که بخش بسیار بزرگی از ساختار شبکه اینترنت شامل سرویس دهنده ها,ایستگاههای کاری (workstation)وماشین نهایی(Hosts)به نحوی از یونیکس استفاده می کنند.

توجه داشته باشید که وقتی از یونیکس صحبت میکنیم بطور عام منظورمان تمام سیستم عاملی است که به نحوی سازگار با ان هستند و همه از اصول مشترکی تبعیت میکنند.تعداد این سیستم عاملها کم نیست و در ادامه به بعضی از انها اشاره خواهیم کرد.

در سالیان اخیر طراحان سیستم عاملی مثل لینوکس,OpenBSD کدهای برنامه این سیستم ها را به رایگان در اختیار عموم قرار داده اند و این امر باعث شکل گیری یک همکاری و نشاط جهانی در توشعه ی آن شد ا,به گونه ای که این سیستم عامل به کامپیوترهای شخصی در خانه ها نیز راه یافت.

عرضه شدن کدهای این سیستم عامل محاسن دیگری نیز داشت که شامل پالایش این سیستم و بهینه شدن عملکرد اجزای مختلف آن و ارائه شدن ابزارهای بسیار قدرتمند و ارزان برای آن بود.از طرفی شفاف بودن عملکرد آن,قابلیت اعتماد وامنیت ویژه ای برای این سیستم عامل پدید آورده است.بسیاری از شرکت های بزرگ مثل IBMکه خود سابقه ای طولانی در توسعه ی سیستم عامل داشتند بالاخره متقاعد شدند که برای کاهش هزینه های خود به این سیستم عامل روی آورند.حمایت این شرکتها با توجه به سرمایه ی فکری و تکنولوژیک و همچنین پشتوانه ی اقتصادیشان باعث شکوفایی و قدرتمند شدن سیستم عامل یونیکس شده است.گذشته از قدرت و امنیت بالا,ارزان بودن یونیکس و ابزارهای مبتنی بر آن,عامل ویژه ای در جذب شرکت ها,موسسات و دانشگاه ها به سوی این سیستم عامل بوده است.

اگر چه یونیکس زیبا و قدرتمند است ولی به دو دلیل همانند یک غول به نظر میرسد:

اول اینکه سیستم عامل واحدی به نام یونیکس وجود ندارد بلکه دهها گونه ی سازگار با آن نوشته شده است و توسعه ی آن در انحصار شرکت یا گروه خاصی نیست;لذا هرگونه,در عین سازگاری با یونیکس قابلیتها و ویژگی خاص خود را دارد.

دوم آنکه استفاده ازآن فقط از متخصصین حرفه ای و مجرب بر می آید.یونیکس دوست مهربان حرفه ای هااست در حالی که به آماتورها و کاربران عادی بسیار سخت گیر است و خشم آنها را بر خواهد انگیخت.هرچند در نسخه های اخیر سیستم عامل های سازگار با یونیکس (مثل لینوکس)سعی شده دل کاربران عادی هم به دست آورد ولیکن هنوز در ذهن بسیاری از افراد کار با یونیکس کابوسی بیش نیست!!

برخی از سیستمهای عامل سازگار با یونیکس در زیر معرفی شده اند:

- Solarisمحصول شرکت Sun Microsystems
- IRIXمحصول شرکت Silicon Graphics
- AIXمحصول شرکت IBM
- HP-UXمحصول شرکت Hewlett Packard
- SCO-UNIXمحصول شرکت Santa Cruz Operation,Inc
- FreeBSDنسخه رایگان یونیکس محصول دانشگاه برکلی آمریکا
- OpenBSDمحصول دانشگاه برکلی به همراه کدهای برنامه آن.این سیستم عامل با شعار "تلاش برای امن ترین سیستم عامل دنیا" ارائه شده است.
- Linuxسیستم عامی است به همراه کدهای رایگان که ابتدا توسط یک دانشجوی فنلاندی بنام لینوس ترووالدنوشته شد و بعدا توسط شرکتهای مختلفی(به قیمت ناچیز)توسعه یافت.اسامی شرکتهائی که نسخه های مختلف لینوکس را تولیدوعرضه کرده اند عبارتند از:

Redhat	Caldera
Corel	Debian
Slackware	Strom
SuSE	YellowDog
Turbo Linux	Mandarke

SunOsنسخه ی قدیمی تر یونیکس که قبل از سولاریس توسط شرکت Sun Microsystemsتهیه شده بود و هنوز در برخی از محیط ها استفاده می شود.
تمام این نسخه های متعدد,عظیم و جهانی از تفکر افرادی نشات گرفته اند که سی سال پیش در شرکت AT&Tو در سکوت بدون حمایت های جهانی اقدام به پیاده سازی یونیکس کردند.بسیاری از انهاامروزه در دوران کهولت هستند و آان گذشته روشن را به دست فراموشی سپردندوبرخی نیز در نقش پیشکسوت وپدران معنوی هنوز در پالایش و رشد آن میکوشند.
سیستم مدیریت فایل,فراخوانی روالهای سیستمی,فرامین وبرخی از گزینه های پیکربندی در نسخه های گوناگون یونیکس به روش های متفاوتی پیاده سازی شده اند که این تفاوتها ناشی از نیاز تکنولوژیک روز بوده است;ولی در مجموع خطوط اصلی این سیستم عامل توسط دو موسسه تعیین میشود:

- گروه توسعه ی یونیکس در شرکت AT&T
- گروه توسعه ی یونیکس BSDدر دانشگاه برکلی

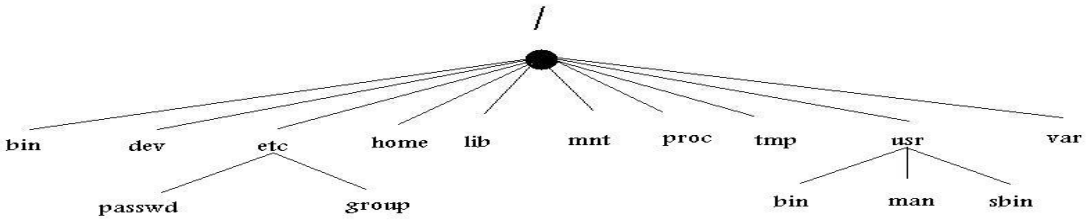
سولاریس, HP-UX از خطوط تعیین شده توسط AT&Tتبعیت میکنند در حالی که FreeBSD,OpenBSDولینوکس به گروه توسعه ی BSDگرایش دارند.سیستمهای عامل AIXو IRIXتلفیقی از نکات برجسته ی خط مشی هر دو گروه هستند لذا اندکی پیچیده تر به نظر میرسند.
در این مبحث مواردی را بررسی میکنیم که تقریبا در تمام نسخه های سازگار با یونیکس مشابه است و مبنای سازگاری همه ی انها با یکدیگر میباشد.لذا وقتی در این مقاله از یونیکس نام می بریم تمام نسخه های سازگار با آن را نیز شامل می شودووقتی حمله ای بر علیه یونیکس تجزیه و تحلیل میشود تمام نسخه های مختلف نسبت به آن حمله در معرض خطر هستندمگر آنکه صراحتا نوع نسخه سیستم عامل را عنوان کرده باشیم.
یونیکس محیط دوستانه و مهربانی برای کاربران عادی نیست بلکه محیطی امن و قدرتمند برای حرفه ای ها محسوب میشود لذا معمولا نفوذگران برای حمله از این سیستم عامل استفاده میکنند هر چند ممکن است ماشین مورد حمله سیستم عاملی مثل ویندوز داشته باشد.
از طرفی سیستم عامل یونیکس,خود,مجموعه کاملی از مستندات لازم در ارتباط با فرامین و برنامه ها را عرضه کرده است.در خط فرمان با دستور Manبسادگی این مستندات را خواهید یافت:

```
$ man [System command]
```

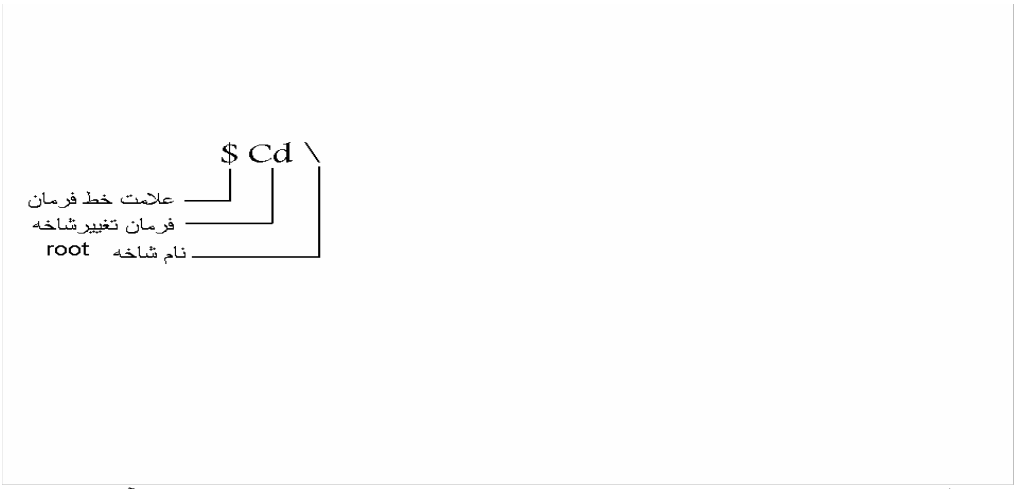
معماری یونیکس: ساختار سیستم فایل

در سیستم عامل یونیکس هر چیزی بصورت یک فایل تلقی ومدل می شود.مثلا حتی دستگاههای جانبی مانند چاپگر یا پورت های مخابره ی داده در قالب یک فایل مدل شده اند و برنامه نویس وقتی می خواهد اطلاعاتی را چاپ کند یا آنها را به منظورمخابره روی پورت Comبفرستد بسادگی فایل استاندارد ان را باز کرده,درون آن را مینویسد و سپس فایل را می بندد.این یکی از مفاهیم عالی یونیکس است که در سطح برنامه نویسی,برنامه نویس درگیر جزئییات سخت افزاری و نرم افزاری این ابزارها نخواهد شد بلکه این جزئیات

در سطح سیستم عامل حل و فصل می شود.عناصر پروسه ها,صفها و بافرها نیز بصورت فایل مدل سازی شده اند.
سیستم عمومی فایل در یونیکس ساختاری درختی همانند شکل زیر دارد .این ساختار با اندکی اختلاف در انواع مختلف یونیکس پیاده شده است.



بگونه ای که در شکل بالا ملاحظه میکنید در راس سیستم فایل یونیکس شاخه root قرار دلرد که برای سادگی شاخه "/" نامگذاری شده است وتمام شاخه ها و فایل ها را در بر میگیرد.تمام زیر شاخه های دیگر (Subdirectories)درون این شاخه قرار میگیرند وکاربران فقط به برخی از آنها دسترسی خواهند داشت.عالیترین سطح دسترسی به سیستم فایل در اختیار کاربری است به شاخه ریشه(با تمام زیر شاخه های آن)تسلط دارد.در محیط یونیکس اگر چنین مجوزی داشته باشید با تغییر شاخه به شاخه / در راس سیستم فایل قرار خواهید گرفت که این کار با اجرای دستور زیر امکان پذیر است.



هر فایل درون یکی ازشاخه های این ساختار درختی ذخیره میشود و برای دسترسی به آن به غیر از نام فایل باید سلسله مراتب شاخه های آن نیز مشخص باشد.مثلا اگر فایلی با نام 01.txt داشته باشید که درشاخه Usrذخیره شده با آدرس usr/01.txt/ قابل شناسایی و ذخیره وبازیابی است.
سیستم عامل یونیکس برای خود یک فرهنگ و ادبیات ویژه وضع کرده است ولذا بسیاری از توسعه دهندگان نسخه های سازگار با آن از فرهنگ حاکم بریونیکس تبعیت کرده اند.بعنوان مثال شاخه هایی که در سطح اول از سیستم فایل ودرون شاخه ریشه(root) تعریف می شوند تقریبا نامهای مشابهی دارندوفایلهای درون آن نیز در ارتباط با یک هدف خاص ذخیره می شوند.در ادامه مهمترین شاخه های تعریف شده در سطح ریشه را معرفی میکنیم:
شاخه /: شاخه ریشه بالاترین سطح در سیستم فایل است وتمام شاخه ها و فایلها درون آن ذخیره شده اند.
برخی از زیر شاخه های عمومی این شاخه عبارتند از:

شاخه /bin:(یا sbin/ در برخی از نسخه ها):این شاخه در بر گیرنده فایلهای اجرائی پایه ای است که وجود آنها برای بوت شدن سیستم الزامی میباشد.

شاخه /dev: این شاخه در بر گیرنده فایل هایی است که دستگاه های جانبی مانند ترمینال,مودم,دیسک های سخت, cd و دیگر سخت افزارهارا راه اندازی ومدل سازی میکنند.درحقیقت فایل های این شاخه "راه انداز ابزار یا Devire Driver"هستند وبرای استفاده از هر دستگاه جانبی متصل به سیستم لازمند.

شاخه /etc: در این شاخه فایل های پیکر بندی سیستم (configuration) فایل تعریف کلمات عبور (password files)فایل تعریف اسامی و آدرسهای شبکه و فایل تنظیمات بوت(boot Setting) ذخیره می شود

شاخه /home: در این شاخه, برای هر کاربر یک زیر شاخه شخصی ساخته می شود.هر کاربر در محیط یونیکس باید حد اقل یک شاخه برای ذخیره وبازیابی اطلاعات شخصی خود داشته باشد.این زیر شاخه بطور معمول در شاخه home/تعریف و ایجاد می شود.

شاخه /mnt : این شاخه محل ذخیره ی فایل هایی است که قادرند یک سیستم فایل دیگر را بطور موقت به درون سیستم فایل اصلی یونیکس Mountکنند. بعنوان مثال اگر در یک ماشین با سیستم عامل یونیکس,یک دستگاه (CD Drive) وجود داشته باشد تمام فایل های درون cd در قالب یک شاخه به درون سیستم فایل اصلی نگاشته می شود. یعنی یونیکس یک CD Drive را با تمام فایلها و شاخه های درون آن بصورت یک شاخه مستقل در درون سیستم فایل اصلی منتقل می کند.مثلا اگر نام این شاخه /mnt/cdباشد با تغییر مسیر به این شاخه در حقیقت به فایل ها وشاخه های CDدسترسی خواهید داشت.

شاخه /proc: تصویر حافظه (memory Image)از پروسه هایی که در حال حاضر بر روی سیستم در حال اجرا هستند.

شاخه /tmp: این شاخه در بر گیرنده فایلهایی است که به هر دلیل در حین راه اندازی سیستم بطور موقتی ایجاد می شود وهنگام خاتمه ی کار سیستم عامل(shutdown)حذف خواهند شد.

شاخه /usr: در این شاخه مجموعه ای از فایل های حیاتی و بنیادی برای سرویس دهی به کاربران ذخیره می شود. به طور معمول در شاخه usr/زیر شاخه های زیر ایجاد می گردد:

شاخه /usr/bin: در بر گیرنده فایل های سودمند و لازم سیستمی است.تمام این فایل ها اجرائی هستند

شاخه /usr/man: شامل فایل های مربوط به کتابخانه ی زبان C (C library files)

شاخه /usr/sbin: برنامه های اجرایی مفید برای عملیات مدیریت سیستم

شاخه /var: در بر گیرنده فایل هایی است که به طور متناوب تغییر می کنند وبرای اهداف مدیریتی سودمند هستند. بعنوان مثال فایل های ثبت وقایع وعملکرد(log files)در شاخه var/log/ذخیره می شوند.یا فضای موقت دیسک برای سرویس دهنده هایی مثل چاپگر یا پست الکترونیکی در این شاخه تعریف می شود.

در سیستم عامل یونیکس دو شاخه مهم دیگر به نامهای "." و ".." تعریف شده است که معنای خاص ومهمی دارند:
شاخه "." : این شاخه نام مستعار شاخه جاری است که کاربر درون آن قرار دارد.

شاخه ".." : این شاخه نام مستعار شاخه پدری شاخه فعلی است.

به عنوان مثال فرض کنید شما به شاخه etc/ تغییر مسیر داده اید و آنرا شاخه جاری تعیین کرده اید.حال اگر فرمان :

$ ls – a

رااجرا نمایید لیست تمام فایل های و زیر شاخه های درون این شاخه را خواهید دید.(دستور ls از سیستم عامل می خواهد تا فهرست فایل های شاخه جاری را نشان بدهد وگزینه a–بدین معناست که سیستم عامل محتویات شاخه را به طور کامل وجامع فهرست کند.) پس از اجرای فرمان ls –a در فهرست نشان داده شده روی خروجی نام شاخه های "." و ".." را خواهید دید."."به نام شاخه جاری یعنی etc/اشاره دارد. یعنی شما می توانید یک فایل را با دو نام زیر احضار نمائید:

"./filename"
"/etc/filename"

نام".." به شاخه پدری که دقیقا در یک سطح بالاترقرار گرفته است اشاره دارد.یعنی اگر شما در شاخه ای مانند /usr/bin/cpp باشید,نام .. به شاخه /usr/bin اشاره می کند. یعنی دو دستور زیر معدل هم هستند:

$ cd /usr/bin
$cd ..

معماری یونیکس:هسته سیستم عامل و پروسه ها

یونیکس از ساختار پیمانه ای(ماجولار) استفاده کرده است. بگونه ای که در پایین ترین سطح یک هسته با بستر سخت افزاردرگیر است وتمام برنامه هابصورت هویتی مستقل حول هسته شکل می گیرند.در یونیکس هسته خودش یک برنامه اجرایی است که قبل از هر برنامه دیگراجرا می شود وکنترل کل سخت افزاررا به دست می گیرد ودسترسی پروسه ها به سخت افزار فقط از طریق هسته ممکن است.این هسته در سیستم عامل یونیکس Kernel نامیده شده است.هسته,قلب سیستم عامل محسوب می شودوهیچ برنامه ای بدون آن نخواهد توانست از مولفه های سخت افزاری سیستم استفاده کند.

برای شروع اجرای یک برنامه در محیط یونیکس,هسته یک پروسه ایجاد می نماید. پروسه برنامه ای در حال اجراست که دو قسمت زیر را در بر می گیرد:

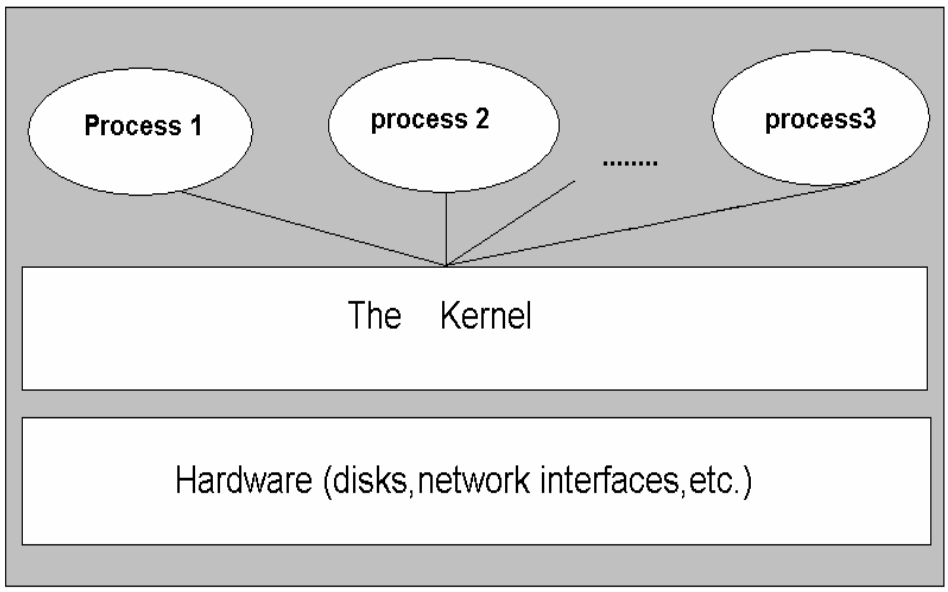
- کدهای اجرایی برنامه
- حافظه اختصاص داده شده به آن برنامه به عنوان فضای کاری(workspace)(تفاوت مفهوم برنامه -program- با پروسه-process-از همین نکته ناشی می شود.)

برنامه در حال اجرا مثل سرویس دهنده وب وپست الکترونیکی که فضا ومنابع لازم را در اختیار دارند"پروسه" نامیده می شوند.پروسه ها را به صورت حباب هایی فرض کنید که بر روی هسته سیستم عامل تشکیل می شوندوتمامی هستی و بضاعت خود را از سیستم عامل دارندوهرگونه اختلال در هسته آنها را نابود خواهد کرد.هسته با ایجاد پروسه ها این حباب ها را خلق می کند و بر عملکرد آنها نظارت دارد.CPU را در اختیار آنها قرار می دهد یا به سادگی آن ها را نابود می کند.

محیط یونیکس محیطی "چند وظیفه ای یا Multitask"است و بالطبع در هرلحظه فقط یکی ازپروسه ها, CPU را در اختیار دارد.پروسه های مختلف به صورت اشتراک زمانی(Time Sharing) از CPU استفاده می کنندوهسته به کمک سخت افزار,زمان دراختیار گرفتن CPU توسط یک پروسه را تنظیم وکنترل می نماید.

هسته سیستم عامل همچنین فضای حافظه مورد نیاز را در اختیار پروسه ها قرار می دهدوباز هم به کمک سخت افزار جلوی تخطی پروسه ها از حریم فضای حافظه اختصاص داده شده به هر پروسه ,از دسترس پروسه های دیگر مراقبت خواهد شد و هرپروسه ای که سعی کند در خارج از فضای حافظه خود چیزی بنویسد یا بخواند توسط هسته سیستم عامل متوقف(ونابود)خواهد شد.

در شکل زیر شمای کلی سیستم عامل وپروسه ها را نشان می دهد.در این شکل هسته بر روی سخت افزار قرار گرفته وپروسه ها بر روی هسته تشکیل می شوند.



دایمون چیست؟

برخی از پروسه ها در سیستم عامل یونیکس در پس زمینه(background)اجرا می شوند ونقش های کلیدی در عملکرد سیستم بر عهده دارند.بعنوان مثال ارائه سرویس های شبکه شبیه سرویس اشتراک فایل (file Sharing) دسترسی به وب وسرویس های چاپ توسط این پروسه ها در اختیار پروسه های دیگر قرار می گیرند.این پروسه ها پس زمینه که بخشی از سیستم عامل محسوب می شوند ولی خودشان بعنوان یک پروسه از هسته استفاده می کنند دایمون -daemon- نامیده می شوند.

دایمون ها معمولا بر اساس سرویسی که ارائه می کنند بصورت زیر نامگذاری می شوند:

"ServiceName+**d**"

مثل: Telnet**d** , ftp**d** , http**d** ,...

پروسه هائی که به طور خودکار در حین راه اندازی سیستم اجرا خواهند شد:

Cron و Inetd و Init

تمام پروسه هایی که بعنوان سرویس دهنده های کلیدی سیستم اجرا می شوند از عمومیترین آنها یعنی سرویس دهنده های وب تا کم کاربرد ترین آنها (مثل: Character Generator) باید توسط پروسه های دیگر راه اندازی و فعال شوند. یعنی پروسه های سرویس دهنده به صورت مستقل وقائم به خوداجرا نخواهند شد بلکه پروسه های کلیدی دیگری آنها را راه اندازی خواهد کرد. بعبارت دیگر هسته خود را درگیر اجرای پروسه های مختلف نمی کند بلکه چند پروسه مدیریتی و کلیدی را اجرا کرده ووظیفه فعال سازی واجرای پروسه های سرویس دهنده را به این پروسه کلیدی محول می نماید.این پروسه کلیدی که در عین راه اندازی سیستم اجرا می شوند به شرح ذیل هستند:

پروسه Init:

این دایمون معمولا اولین پروسه ای است که پس از هسته شروع به کارمی کند وپدر تمام پروسه های کاربردی است که پس از راه اندازی سیستم,اجرا می شوند. این پروسه پس از بوت شدن سیستم,اسکریپت های پیکر بندی را اجرا می نماید تا راه اندازی سیستم با اجراشدن پروسه های لازم تکمیل گردد.

محل قرار گرفتن اسکریپت های پیکر بندی سیستم,به طور معمول **/etc/rc.d** , **/etc/init.d** می باشد.
بر اساس این فایل ها پروسه های ثبت وقایع (logger) , برنامه ی زمانبندی وپروسه های راه انداز واسط های شبکه (Network Interface) پیکر بندی واجرا می شوند.
همچنین init , متولی راه اندازی پروسه ی سرویس دهنده های مهمی است که پس از اجرا به صورت دایمون به یک شماره پورت خاص گوش می دهند, ترافیک بسته های TCP , UDP را دریافت وپردازش می کنند وبا کاربران در تعامل هستند.
برخی از این دایمون ها عبارتند از:

- **Httpd:** سرویس دهنده ی وب که تقاضا های HTTP و HTTPS را پردازش می کندو به طور معمول به پورت شماره ۸۰ گوش می دهد.
- **Sendmail:** سرویس دهنده ی پست الکترونیکی در محیط یونیکس که به پورت شماره ۲۵ گوش می دهد

- **NFS:** پروسه سرویس دهنده ای که در محیط یونیکس فایل ها را به اشتراک می گذارد.
(network Files System) این سرویس دهنده در ابتدا توسط شرکت Sun Microsystems ابداع شد ولی بعدا تمام سیستم های عامل سازگار با یونیکس از آن بهره گرفتند.

در ادامه این سه سرویس دهنده را اندکی بیشتر بررسی خواهیم کرد.این پروسه ها پس از اجرا,به شماره پورت مربوطه گوش داده به حالت انتظار فرو می روند تا زمانی که تقاضای یک ارتباط دریافت شود یا ترافیکی به آن شماره پورت وارد گردد.دقت کنید که سه سرویس دهنده ی فوق دارای حجم ترافیک ورودی بسیار بالایی هستند و باید بطور دائم در حافظه آماده پردازش این ترافیک باشند.لذا احضار آنها از روی سیستم فایل تاخیر زیادی به سیستم تحمیل خواهد کردبه همین دلیل این پروسه ها ی سرویس دهنده , به طور مستقل توسط init در حین راه اندازی سیستم اجرا خواهند شدو به طور ثابت و دائم در حافظه باقی خواهند ماند.
از طرف دیگر سرویس های دیگری مثل TelNet یا FTP بطور مداوم مورد استفاده قرار نمی گیرند و ترافیک زیادی ندارند لذا بصورت مستقل و دائم به حافظه بار نخواهند شد.اگر تمام پروسه های سرویس دهنده ای که به یک شماره پورت خاص گوش می دهند,به درون حافظه بار شوند و بطور مستقل و دائم به حالت انتظار بروند,کارایی سیستم را کاهش داده و منابع آن بالاخص حافظه را تلف خواهند کرد.
برای افزایش کارایی و صرفه جویی در منابع سیستم به غیر از پروسه های Httpd , NFS, Sendmail که مستقیما توسط init اجرا می شوند,هیچ پروسه ای نمی تواند بطور مستقل به یک شماره پورت گوش بدهد.
در یونیکس پروسه ای به نام inetd (Internet Daemon) وجود دارد که به تمام شماره پورت های TCP یاUDP گوش می دهد و در هنگام ورود ترافیک به هر یک از پورت ها پروسه های متناظر باآن را فراخوانی کرده و اجرا می نماید.

پروسه Inetd:

دایمون Inetdدر مرحله راه اندازی سیستم توسط پروسه ی init (که شرح آن را در بالا خواندید) به همراه سه پروسه Sendmail , NFS , Httpd اجرا خواهد شد.تنظیمات این پروسه ی حیاتی در فایلی با نام /etc/services تعیین و پیکر بندی می شود.
وقتی ترافیکی به ماشین وارد می شود, inetd ابتدا از طریق تنظیمات inetd.conf سرویس مربوطه را مشخص کرده و سپس بر اساس فایل /etc/services پروسه ی متناظر با آن سرویس را راه اندازی و اجرا می نماید.پروسه ی سرویس دهنده پس از اجرا, سرویس لازم را ارائه کرده و سپس خاتمه می یابد.در حقیقت به جای آنکه پروسه های گوناگون به پورت ها گوش بدهد, inetd به نیابت از همه آنها به تمام پورت های تعیین شده گوش خواهد دادودر هنگام لزوم آنها را راه اندازی و اجرا خواهد کرد. بطور معمول inetd سرویس زیر را راه اندازی و اجرا می نماید:

- **Echo:** این سرویس دهنده هر رشته کاراکتری را که دریافت کند عینا به سوی مبدا آن باز می گرداند.این سرویس برای اشکال زدائی از شبکه مورد استفاده قرار می گیرد

- **Chargen:** این سرویس دهنده یک فهرست از کاراکتر ها را تولید و برای مبدا ارتباط بر می گرداند. بطور معمول هدف این سرویس دهنده اندازه گیری کارائی و سرعت یک سیستم در شبکه می باشد.حذف این سرویس دهنده چندان مهم نخواهد بود.

- **FTPd:** این دایمون مهم سرویس دهنده ی FTP را راه اندازی می نماید تا کاربران راه دور از طریق آن به مبادله فایل بپردازند.

- **TelNetd:** این دایمون سرویس دهنده یTelNet برای سرویس دهی از راه دور به کاربران است بگونه ای که بتوانند به سیستم وارد شده وفرامین مورد نظر خود را روی ماشین اجرا نمایند.(این سرویس دهنده نیز بسیار خطرناک است واگر یک نفوذگر بتواند به نحوی در قالب یک کاربر مجاز به سیستم وارد شود , امنیت آن سیستم وحتی کل شبکه به خطر خواهد افتاد.)

- **Login وShell:** این دو پروسه نیز به کاربران اجازه می دهند که از راه دور به سیستم وارد شوند (با فرمان rlogin) یا یک نشست با سیستم برقرار کرده و فرامین راه دور خود را از طریق برنامه rsh (remote Shell)اجرا نمایند.

- **TFTP:** این سرویس دهنده ,پروسه ی TFTP را راه اندازی و اجرا می نماید. TFTP امکانات بسیار ساده و مختصری برای انتقال فایل ها دارد.

برای آنکه inetd را مجبور کنید تا یک پروسه ی سرویس دهنده را راه اندازی و اجرا نماید باید یک خط به فایل پیکر بندی inetd اضافه نماییدو در آن نام سرویس وپروسه ی اجرائی متناظر را تعیین کنید. به گونه ای که در این فایل مشاهده می شود برای غیر فعال کردن هر سرویس,کافی است درون فایل inetd.conf به دنبال خط مربوطه بگردید و با یک ویرایشگر ساده ی متن در ابتدای خط علامت # را قرار بدهید.inetd تمام خطوطی که با # شروع شده باشند را خط توضیحی تلقی کرده و نادیده می گیرد.در زیر مثالی از فایل پیکر بندی inetd مشاهده می شود:

```
#These are Standard Services
#
ftp     Stream    tcp     nowait    root     /usr/sbin/in.ftpd     in.ftpd
telnet  Stream    tcp     nowail    root     /usr/sbin/in.telnetd  in.ftpd
#
shell   Stream    tcp     nowail    root     /usr/sbin/in.rshd     in.rshd
login   Stream    tcp     nowail    root     /usr/sbin/in.rlogind  in.rlogind
#exec   Stream    tcp     nowail    root     /usr/sbin/in.rexecd   in.rexecd
```

هرکدام از فیلد ها ی مختلف هر سطر در فایل inetd.conf مشخصه ی ویژه ای را تعیین وتنظیم می کنند که از چپ به راست آنها را معرفی می نماییم:

- **فیلد اول Service Name:** در این فیلد نام سرویسی که باید عرضه شود به صورت نمادین درج می شود. این نام کلیدی خواهد بود برای مراجعه به فایل پیکربندی /etc/services و استخراج تنظیمات سرویس دهنده و اجرای آن; در حقیقت تنظیمات /etc/services شماره پورت هائی که باید inetd به آن گوش بدهد را مشخص می کند.

- **فیلد دوم Socket Type:** این فیلد نوع ارتباط با سرویس دهنده را مشخص می کند.در این فیلد یکی از گزینه های زیر تنظیم می شود:
  - **Stream:** سرویسی مبتنی بر TCP
  - **Dgram:** سرویسی مبتنی بر UDP
  - **Seqpacket:** سوکت های مبتنی بر بسته های شماره گذاری شده
  - **Rdm:** سوکت های مبتنی بر دریافت مطمئن پیام

نکته مهم این است که اگر ارتباط از نوع Stream تعیین شود دلیلی ندارد که لایه ی زیرین آن حتما TCP باشد ولیکن در شبکه ی اینترنت که مبتنی بر TCP/IP است سوکت های نوع Stream قطعا مبتنی بر TCP و سوکت های نوع dgram مبتنی بر UDP است.

- **فیلد سوم Protocol Name:** در این فیلد نام پروتکل لایه ی انتقال درج می شود.در مورد سرویس دهنده های شبکه اینترنت,فقط گزینه های TCP و UDP کاربرد دارند درحالیکه درج گزینه 'rpc/tcp' (مخفف remote procedure call) و گزینه rpc/udp نیز ممکن خواهد بود.

- **فیلد چهارم Wait Status:** در این فیلد دو نوع گزینه ی حیاتی زیر می تواند درج شود:
  - **Wait:** این گزینه بدیم معناست که یک پروسه ی سرویس دهنده قادر است فقط با یکبار اجرا چندین تقاضای همزمان را پردازش کند بعبارت دیگر تمام تقاضاهاتحویل یک پروسه واحد و مستقل خواهد شدولذا لازم نیست به ازای هد تقاضا یک پروسه ی مستقل ومشابه اجرا شود.

  - **Nowait:** این گزینه مشخص می کند که inetd باید به ازای پذیرش هر تقاضا از یک شماره پورت, یک نسخه مستقل از پروسه ی سرویس دهنده رابرای پاسخ به آن تقاضا اجرا کرده و پس از خاتمه ی سرویس آنرا از بین ببرد,لذا با ورود هر تقاضا پروسه از نو تولید و اجرا خواهد شد.

- **فیلد پنجم user name:** در این فیلد سطح دسترسیپروسه ی سرویس دهنده به منابع سیستمی تعیین می گردد,یک پروسه که در سطح root اجرا می شود بالاترین مجوز دسترسی به منابع سیستمی را خواهد داشت ولی هرگاه پروسه در سطح یک کاربر معمولی اجرا شود فقط منابعی را در اختیار خواهد داشت که برای آن کاربر تعیین شده است.

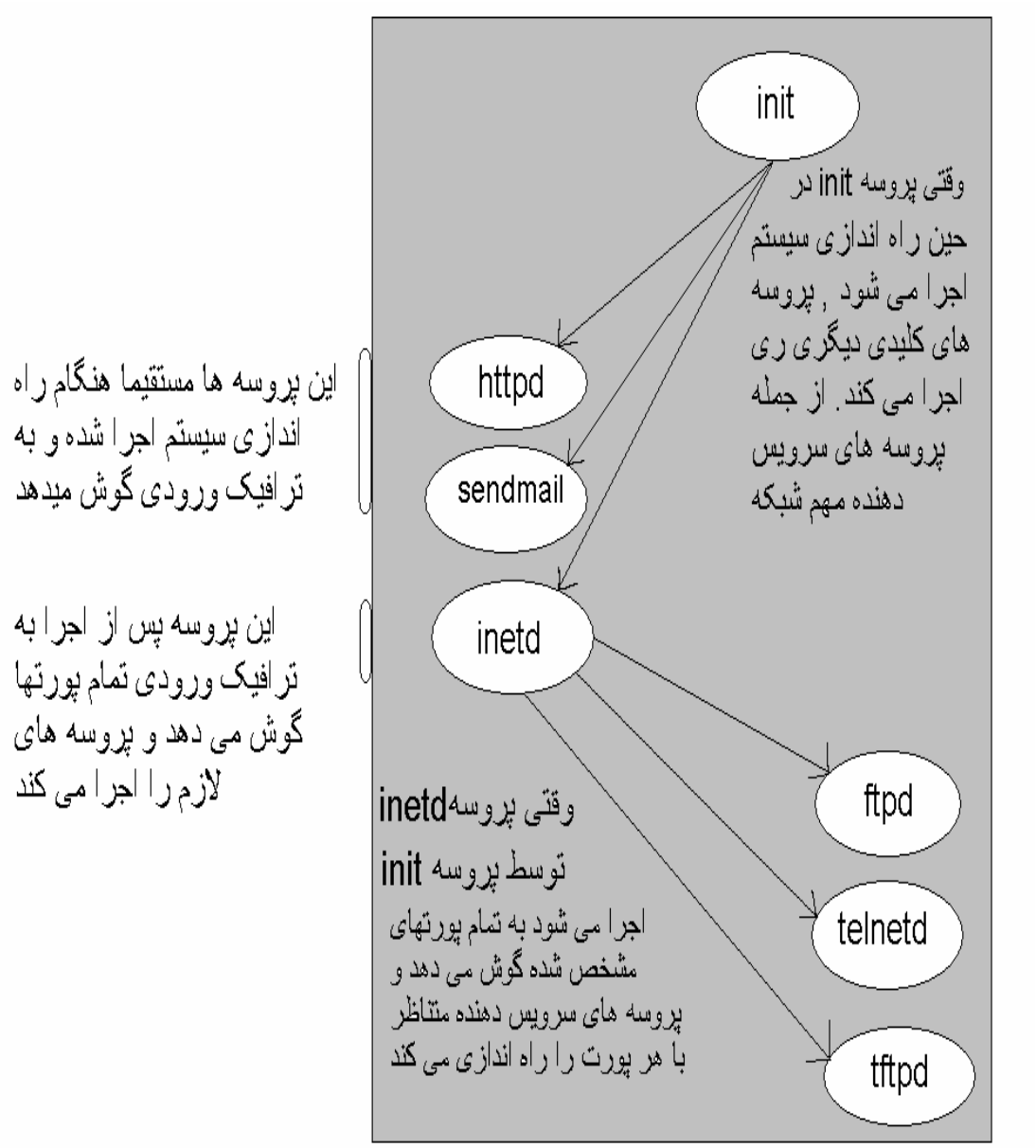
دقت کنید که در سیستم عامل یونیکس, یک کاربر با نام root بالاترین سطح دسترسی را دارد و در حقیقت این کاربر کسی نیست مگر مسئول آن سیستم(یعنی Administrator)

- **فیلد ششم Server Program:** در این فیلد نام برنامه ی اجرایی که برای ارائه سرویس بدان شماره پورت باید اجرا شود,درج می شود.

- **فیلد هفتمServer Program Arguments :** وقتی برنامه ی مشخص شده در فیلد ششم اجرا شدلازم است یکسری از آرگومانهای ورودی جهت شروع به آن ارسال شود.این فیلد فهرست آرگومانها یخط فرمان را که باید به آن برنامه فرستاده شود , تعیین می کند.

ارتباط بین پروسه های inetd , init وپروسه های سرویس دهنده ی شبکه در شکل زیر نمایش داده شده است. بگونه ای که در این شکل می بینید, پروسه init وپروسه های دیگری را اجرا نماید کهمستقلا به شماره پورتهای مشخصی گوش می دهند.از بین این پروسه ها , inetd به مجموعه ای از شماره پورت ها گوش داده و به ازای هر تقاضا, پروسه ی متناظر با آنرا راه اندازی و اجرا می نماید.
پروسه هایی مثلhttpd یا sendmail بطور مستقل ترافیک خود را پردازش می نمایند در حالی که پروسه های دیگری همانند دایمون های tftpd , telnetd , ftpd توسط inetd راه اندازی و اجرا خواهند شد.

(شکل زیر ارتباط بین پروسه های inetd , init و پروسه های سرویس دهنده ی شبکه را نشان می دهد)



پروسه Cron:

یکی دیگر ازپروسه هایی که توسط هسته در حین راه اندازی سیستم فرخوانی واجرا می شود, پروسهCron نام دارد.این پروسه اجرای منظم وزمانبندی شده یکسری از فرامین (درزمان های از قبل تعیین شده)را بر عهده دارد.(بعبارت دیگر این پروسه برای زمانبندی دقیق و منظم اجرای یکسری از پروسه های دیگر در خدمت مسئول سیستم خواهد بود.) بعنوان مثال شاید , مسئول سیستم علاقمند که یک برنامه ویروس یاب در ساعت ۳ بامداد به جستجوی سیستم فایل مشغول شود یا نیمه شب هر شب از اطلاعات سیستم فایل نسخه پشتیبان تهیه گردد; در این حالت او مجبور خواهد بود از پروسه Cronبرای این کار بهره بگیرد.لذا براحتی با تنظیم یک خط در فایل پیکر بندی Corn که Crontab نام دارد هدف خود را دنبال می نماید. بطور معمول در سیستم عامل یونیکس این فایل پیکر بندی در محلهای زیر ذخیره و بازیابی می شود:

/etc/crontab یا /usr/lib/crontab

به همان ترتیب که مسئول سیستم میتواند از Cron استفاده کند یک نفوذگر نیز ممکن است موفق شود اجرای برنامه خود رابه Cronمحول نماید!او پس از رخنه در یک سیستم با تغییر در فایلCrontab فرامین خود را برای اجرا تنظیم و زمانبندی می نماید!به عنوان مثال او می تواند پس از نفوذ به یک سیستم آنرا جهت حمله نوع Dos زمانبندی کند.

شروع و اجرای پروسه هابه صورت دستی

بگونه ای که اشاره شد پروسه های init, inetd , cron بطور خودکار (در حین راه اندازی سیستم) پروسه هایی رو بر روی یک ماشین اجرا می کنند.کاربران سیستم (یا مسئول سیستم) باید بتوانند بصورت دستی پروسه های خود را بر روی یک ماشین اجرا نمایند.هرگاه شما با درج کردن نام یک برنامه در خط فرمان,آنرا اجرا می کنید در حقیقت یک پروسه ایجاد و اجرا شده است.البته مجوز یک پروسه ی متعلق به کاربر برای دسترسی به منابع سیستم در همان سطحی است که مسئول سیستم برای آن کاربر تعیین کرده است.

وقتی نام یک برنامه روی خط فرمان درج شده وکلید Enter فشار داده می شود,سیستم عامل یرای یافتن و اجرای آن درون شاخه های مختلفی به جستجو خواهد پرداخت.شاخه هایی که سیستم عامل درون آنها به جستجوی برنامه ی اجرایی خواهد گشت برای هر کاربر متفاوت بوده و"مسیر جستجو یا "Search Path نام دارد. کاربر می تواند"مسیر پیش فرض جستجو" را با اجرای فرمان زیر ملاحظه کند:

$ echo $PATH

پس از اجرای فرمان فوق پاسخی شبیه به زیر روی خروجی ظاهر خواهد شد:

/usr/local/bin:/bin:/usr/bin:/usr/xllr6/bin

شاخه هایی که سیستم به طور پیش فرض درون آنها به جستجو یک برنامه اجرائی خواهد پرداخت به ترتیب از چپ به راست مشخص شده اند و علامت **:** هر مسیر را از بعدی متمایز خواهد کرد. در مثال بالااگر کاربر نام یک برنامه اجرایی مانند gcc را در خط فرمان درج کند وکلید enter را فشار بدهد, سیستم عاملابتدا به سراغ شاخه **/usr/local/bin** رفته و در آنجا به جستجوی gcc می پردازد.در صورتی که در آنجا چنین فایلی یافت نشد به سراغ شاخه /bin می رود و در ادامه به ترتیب شاخه های **/usr/bin** و **/usr/xllr6** جستجو می شوند.

شاخه "**.**" (شاخه جاری کاربر)نباید در "مسیر جستجو ی برنامه" وجود داشته باشد چرا که این امر اندکی خطرناک است.فرض کنید که شمافرمان ls را درج و اجرا کرده اید.این برنامه اجرایی باید از شاخه اصلی خودش اجرا شودولی اگر شاخه "**.**" به ابتدای مسیر جستجو اضافه کرده باشیمجستجو از شاخه جاری خودتان شروع خوهد شد.حال اگر فایلی با نام ls در شاخه جاری شما وجود داشته باشدآن فایل اجرا خواهد شد.اگر نفوذ گربه نحوی برنامه ی آلوده را همنام با فرامین اصلی یونیکس روی شاخه جاری شما (که خواندنی/ نوشتنی) است ذخیره کرده باشد بدین نحو به هدف خود خواهد رسید.این برنامه ممکن است یک اسب تراوا یا یک برنامه استراق سمع وجمع آوری کلمات عبور یا برنامه ای برای حمله Dos باشد.

فعل و انفعال با پروسه ها

هرگاه هسته سیستم عامل یک پروسه را اجرا می کند به آن یک شماره ی شناسایی یکتا به نام PID که همان (Process Identifier) انشعاب می دهد.این شماره ملاک شناسائی آن پروسه ومراجعه به آن خواهد بود.کاربر می تواند با اجرای فرمان ps فهرست تمام پروسه های در حال اجرا را روی خروجی مشاهده کند.این فرمان pid هر پروسه , نام برنامه, میزان استفاده از زمان CPU وبرخی از مشخصات دیگر را به کاربر نشان می دهد.

در زیر خروجی فرمان ps در محیط لینوکس نشان داده شده است.(البته برخی خطوط خروجی حذف شده اند و برای خوانا تر شدن برخی از کلمات پررنگ تر شده اند)در این فهرست نام پروسه های , crond , init inetd به وضوح دیده می شود. در این فهرست نام برنامه bash دیده می شود که همان پروسه ی " پوسته فرمان" است که فرمان ps را اجرا کرده است.

 # ps –aux

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
Root	1	0.2	0.7	1120	476	?	S	22:13	0:04	**init**[3]
Root	2	0.0	0.0	0	0	?	SW	22:13	0:00	[kflushd]
Root	3	1.1	0.0	0	0	?	SW	22:13	0:19	[kupdate]
Root	4	0.0	0.0	0	0	?	SW	22:13	0:00	[kpiod]
Root	5	0.0	0.0	0	0	?	SW	22:13	0:00	[kswapd]
Root	6	0.0	0.0	0	0	?	SW<	22:13	0:00	mdrecoveryd
Bin	288	0.0	0.6	1212	420	?	S	22:13	0:00	portmap
Root	303	0.0	0.0	0	0	?	SW	22:13	0:04	[lockd]
Root	304	0.0	0.0	0	0	?	SW	22:13	0:00	[rpciod]
Root	433	0.0	0.9	1328	620	?	S	22:13	0:00	**crond**
Root	462	0.0	0.8	1156	520	?	S	22:13	0:00	**inetd**
Root	995	3.5	1.5	1736	976	pts/0	S	22:46	0:00	bash
Root	1005	0.0	1.3	2504	820	pts/0	R	22:46	0:00	ps -aux

روش فعل و انفعال با یک پروسه ارسال یک Signal برای آن پروسه می باشد.سیگنال نام یک پیام است که روند اجرای پروسه را متوقف کرده و به آن می گوید که باید چه کاری انجام بدهد.یکی از معمولی ترین سیگنال ها سیگنال HUPکه همان(Hang-up) است که یک پروسه مثل inetd را وادار می کند تا از نو فایل پیکر بندی خود را باز خوانی کرده و مجددا بر اساس آن تنظیم شود.
کاربر می تواند با فرمان Kill سیگنال مورد نظر خود را برای یک پروسه ارسال کند در این فرمان ملاک شناسائی پروسه PID آن است, فرمان KillAll همین عملکرد را دارد با این تفاوت که به جای PID شماره شنا سائی از نام پروسه استفاده می شود. Kill و Killall بر خلاف معنای اسمشان فقط برای خاتمه دادن و نابود کردن پروسه ها بکار نمی روند بلکه می توانند یک "سیگنال به پروسه " ارسال کنند. بعنوان مثال فرض کنید مسئول سیستم یا یک نفوذگر در فایل پیکر بندی inetd تغییری ایجاد کرده باشد. برای اعمال این تغییرات لازم نیست که سیستم از نو راه اندازی شود بلکه کافی است تا یکی از فرامین زیر اجرا شود:

$ Kill –hup 462
$Killall –hup inetd

فراموش نکنید Killall به جای Pid نام پروسه را می پذیرد;البته فرمان Killall نامی نا متناسب دارد زیرا که نه تنها تمام پروسه ها رو خاتمه نمی دهد بلکه "فقط سیگنالی خاص را برای یک پروسه"ارسال می نماید.

حساب های کاربری و گروه ها (Acounts & Group)

برای ورود به یونیکس هر کاربر باید یک حساب کاربری بر روی سیستم داشته باشد.هر پروسه نیز در یک سطح از دسترسی اجرا می شود و کاربر بر اساس مجوز های آن سطح , به منابع سیستمی دسترسی خواهد داشت.بدون داشتن یک حساب کاربری کسی قادر نخواهد بود به یونیکس وارد شود و سرویس بگیرد.حال باید دید که چگونه حساب های کاربری در سیستم عامل یونیکس تعیین و تنظیم می شود.

فایل /etc/passwd:

حساب های کاربری در سیستم عامل یونیکس در فایل /etc/passwd تعیین و تنظیم می شود.در این فایل به ازای هر حساب کاربری یک سطر درج شده که مشخصات بسیار مهمی از هر کار بر تعیین می نماید.در زیر مثالی از محتویات فایل /etc/passwd آورده شده است:

Root:1sumys0Ch$ao01LX5MF6u/53/85b3s5raD/ :0 :0 :root: /root: /bin/bash
Bin:*:1:1:bin: /bin:
Daemon:*:2:2:daemon: /sbin:
ftp:*:14:50:FTP User:/home/ftp:
nobody:*:99:99:Nobody:/:
alice:1hwqqWpmr$TeFJcr9xiaMILQmzU9LW.0:501:501:Alice T.
:/home/users/alice:/bin/bash

(به دلیل طولانی بودن فایل بقیه خطوط آن حذف شده است)

هر خط در فایل فوق دارای پارامترهایی است که توسط علامت : از هم تفکیک شده اند این پارامترها از سمت چپ به راست به شرح زیر می باشند.

- **Login name:** در این فیلد نام حساب کاربری یا به تعبیری User Id درج خواهد شد. کاربر باید این نام را در هنگام Login وارد نماید این نام محرمانه نیست وممکن است همه آن را بدانند.

- **Encrypted/Hashed Password:** در این فیلد کلمه عبور کاربر بصورت رمز نگاری شده درج می شود.لذا حتی اگ این فایل در اختیار عموم قرار بگیرد ,استخراج کلمه عبور به راحتی امکان پذیر نخواهد بود. برای رمز نگاری کلمه عبور از روش های متنوعی در یونیکس استفاده می شود. (مثل :Hash Algorithm یا encryption Cipher) وقتی کاربر کلمه عبور خود را وارد می کند الگوریتم رمز نگاری بر روی آن اعمال شده و حالت رمز شده ی آن با فیلد مقایسه می شود.اگر کلمه عبور پس از رمز نگاری با این فیلد یکسان بود به کاربر اجازه ورود داده می شود. در غیر این صورت به او اجازه ورود داده نخوهد شد. اگر در فیلد کلمه عبور کاراکتر"*" درج شود بدین معناست که هیچکس نمی تواند با UserId تعریف شده در فیلد قبلی به سیستم وارد شود.

- **UID Number:** به هر حساب کاربرییک شماره شناسایی یکتا نسبت داده می شود.وقتی یک کاربر به یونیکس وارد می شود این شماره منحصر به فرد ملاک شنا سائی او خواهد بود و سطوح دسترسی پروسه اجرائی او بر اساس این شماره تعیین خواهد شد.

- **Default GID number:** در سیستم عامل یونیکس برخی از کاربران در قالب یک گروه دسته بندی شده و برای همه آنها یک سطح دسترسی به فایل ها و منابع سیستمی در نظر گرفته می شود.در این فیلد شماره گروهی که کاربران بدان متعلق است, درج می شود.(گروه ها در فایل دیگری تعریف می شوند)

- **GECOS Information:** این فیلد با اطلاعات دلخواه کاربر پر می شود و بطور مستقیم مورد نیاز هیچ پروسه ای نیست.این اطلاعات شامل نام و نام خانوادگی یا شماره تلفن او می باشد.

- **Home Directory:** در این فیلد نام شاخه ای درج می شود که کاربر پس از ورود به سیستم در آن شاخه قرار خواهد گرفت. بطور معمول این همان شاخه ای است که برای آن کاربر ایجاد شده و او فایل های محلی خود را در آن ذخیره و باز یابی می نماید.

- **Login Shell:** در این فیلد نام برنامه ای درج می شود که به عنوان "پوسته فرمان -Command Shell-" باید پس از ورود کاربر به سیستم فرامین او را اجرا نماید.

فایل etc/passwd/ فایلی متنی است . هر کاربر یا هر پروسه یا حتی یک نفوذگرقادر است فایل مربوطه را خوانده و کلمات عبور رمز نگاری شده را استخراج نماید.در این صورت نفوذگر بر اساس تکنیک های شکستن (Password Crack) سعی در کشف کلمات عبور کاربران می نماید.لذا در برخی از گونه های یونیکس این فایل بصورت متنی ذخیره نمی شود بلکه کلمات عبور کاربران از درون فایل

/etc/passwd جدا شده و در یک فایل مجزا با نام /etc/shadow یا /etc/secure ذخیره می شوند.لذا کاربران وپروسه ها به تمام فیلد های فایل /etc/passwd دسترسی دارند مگر به فیلد کلمه عبور .هیچ کاربری مگر در سطح root قادر به دسترسی به فایل /etc/shadow نخواهد بود.

فایل /etc/group:

از دیدگاه مدیریت سیستم تعیین مجوز دسترسی برای تک تک کاربران وقت گیری است.برای راحتتر کردن فرآیند تعریف حساب های کاربری , یونیکس اجازه داده است تا گروه های کاربری تعریف شده و برای هر گروه مجوز های دسترسی به دقت تعیین شود.سپس برای تعریف یک حساب کاربری می توان آن را عضوی از یک گروه تعریف کرد و بنا بر این مجوز های دسترسی او دقیقا مشابه تمام اعضای گروه خواهد بود.هر "گروه" در فایل /etc/group تعریف می شود.در این فایل به ازای هر گروه یک سطر وجود دارد که در زیر نمونه ای از محتویات این فایل دیده می شود:

Daemon:x:2:root,bin
Finance:x:25:alice,fred,susan
Hr:x:37:bob,mary

هر خط در این فایل شامل فیلدهایی است که به ترتیب از چپ به راست عبارتند از:

- **Group Name:** در این فیلد نام یک گروه درج می شود

.

- **Encrypted/Hashed Password:** دز این فیلد قاعدتا باید کلمه عبور گروه بصورت رمز نگاری شده درج شود ولی بطور معمول در آن کاراکتر های 'X' یا '*' قرار می گیرد و لذا هیچ کلمه عبوری برای گروه در نظر گرفته نخواهد شد.
- **GID Number:** این فیلد توسط سیستم تنظیم می شود و شماره شناسائی یک گروه محسوب می شود.
- **Group Members:** نام کاربری تمام اعضای گروه در این فیلد درج می شود.هر نام توسط کاراکترهای"," از دیگری جدا می شود

مثلا محتویات نشان داده شده ی فایل /etc/group تعیین می کند که: Alice , Fred, Susan در گروهی با نام Finance تعریف شده اند و دارایGID شماره ۲۵ هستند .

کاربری با نام root

قدرتمند ترین کاربر در سیستم عامل یونیکس با نام کاربری root تعریف شده است که بالاترین سطح دسترسی به منابع سیستم را خوهد داشت.کاربر root می تواند هرفایلی را بخواند, تغییر بدهد یا حذف کند. بدین نحو او قادر خواهد بود پیکر بندی کل سیستم را تنظیم کند, حساب های کاربری یا گروه ایجاد نماید.لذا در برخی از محاورات عمومی, کاربر root با عنوان Super User نامیده می شود . بطور معمول شبکه کاربری (UID) چنین شخصی صفر است.مسئول سیستم تا زمانی که بخواهد تغییراتی در پیکره سیستم عامل ایجاد کند باید با این حساب کاربری به یونیکس وارد شود. هرگاه نفوذ گر بتواند کلمه عبور کاربر root را به دست آورد یا به نحوی در سطح root به سیستم نفوذ کند هرکاری برای او ممکن خواهد بود.

تعیین مجوز های دسترسی در یونیکس

هر فایل در سیستم " مدیریت فایل یونیکس" دارای مجموعه ای از تنظیمات مجوز دسترسی است بدین معنا که چه کسانی حق دارند به آن فایل دسترسی داشته باشند و این دسترسی در چه سطحی است. (خواندن / نوشتن / اجرا) هر فایل در یونیکس یک مالک فردی دارد. یعنی یک شخص با حساب کاربری مشخص صاحب آن فایل است .در ضمن هر فایل دارای یک مالک گروهی یا Group Owner است (یعنی گروهی که اعضای آن می توانند به آن فایل دسترسی داشته باشند.)اما فقط مالک اصلی فایل می تواند مجوز های دسترسی به آن فایل را تغییر بدهد. (به اضافه کاربر root که در عالیترین سطح دسترسی است.) در سیستم فایل یونیکس مجوز های دسترسی به هر فایل در سه دسته مجزا تنظیم می شود:

- مجوز های دسترسی به فایل برای شخص مالک
- مجوزهای دسترسی به فایل برای گروه مالک
- مجوز های دسترسی به فایل برای عموم کاربران سیستم (یعنی تمام کاربران و پروسه با هر حساب کاربری User Account)

برای هرکدام از سه دسته فوق مجوز های زیر قابل تنظیم است:
- اجازه خواندن از فایل
- اجازه نوشتن یا تغییر فایل
- اجازه اجرای آن فایل

این سه مجوز به طور جدا گانه برای هر سه دسته باید تنظیم شود.لذا برای هر فایل باید حداقل ۹مولفه دسترسی توسط مالک آن تنظیم شود:
بگونه ای که اطلاع دارید فرمان ld با گزینهl- فهرست فایلها را به صورت دقیق و کامل روی خروجی نشان می دهد.استفاده از این فرمان با گزینه l- مجوز های دسترسی به هر فایل, مالک فردی فایل و گروه مالک آن نشان خواهد داد. به مثال زیر توجه کنید:

#ls –l

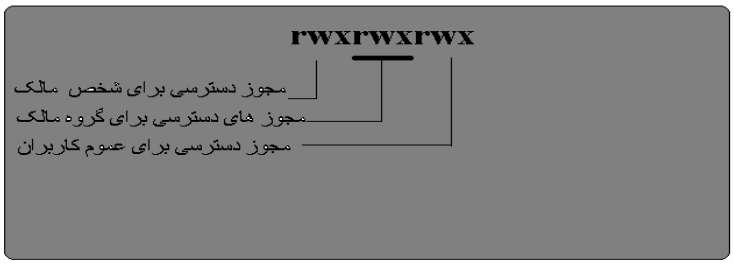
total 1588

drwxr-xr-x	3	root	root	4096	Sep	15	10:17 CORBA
-rw-r--r--	1	root	root	2434	Mar	7	10:17 DIR_COL
-rw-r--r--	1	root	root	4	Mar	11	10:17 Hostname
-rw-r--r--	1	root	root	5472	Mar	1	10:17 Muttrc
drwxr-xr-x	11	root	root	4096	Sep	15	10:17 xll
-rw-r--r--	1	root	root	12	Mar	8	10:17 adjtime
-rw-r--r--	1	root	root	732	Feb	17	10:17 aliases
-rw-r--r--	1	root	root	20480	Sep	15	10:17 aliases.db
-rw-r--r--	1	root	root	370	Mar	3	10:17 anacrontab
-rw-------	1	root	root	1	Mar	1	10:17 at.deny
-rw-r--r--	1	root	root	582	Aug	27	10:17 crontab
drwxr-xr-x	2	root	root	4096	Sep	15	10:28 charsets

(به دلیل طولانی بودن فهرست خروجی بقیه خطوط حذف شده اند.)

دقت کنید که در اولین ستون از فهرست فوق یک الگو با ده کاراکتر دیده می شود که در حقیقت همان مجوز های دسترسی به فایل است.اگر اولین کاراکتر از این الگو ,کاراکتر 'd' باشد, مشخص کننده آن است که مشخصات ارائه شده در آن سطر مربوط به یک شاخه (directory)است نه یک فایل.

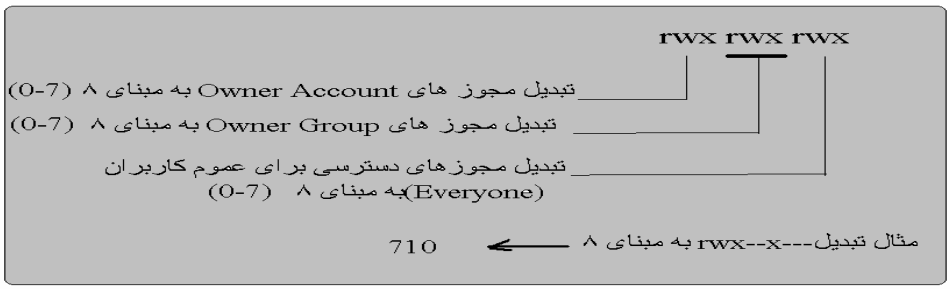
نه کاراکتر بعدی مجوز های دسترسی را برای آن فایل یا شاخه تعیین می کند.الگوی این نه کاراکتر بصورت زیر است:



هرگاه مالک فایل اجازه خواندن فایل را صادر کرده باشد, در خروجی فرمان ls –l کاراکتر'r' ظاهر خواهد شد.بهمین صورت برای صدور مجوز نوشتن کاراکتر'w' و برای مجوز اجرا کاراکتر 'x' در محل تعیین شده قرار می گیرد وگرنه علامت '-' در محل مربوطه درج می شود.
در فهرست مثال بالا دیده می شود که فایل crontabدارای مجوز بصورت '-rw-r--r--' است پس:

- اولا یک فایل است نه یک شاخه, چون در اولین کاراکتر الگو علامت '-' قرار گرفته است
- ثانیا قابل خواندن و نوشتن برای مالک اصلی آن (root) است چون کاراکتر ها rw تنظیم شده اند.
- ثالثا قابل اجرا نخواهند بود چون به جای x کاراکتر '-' قرار گرفته است.
- برای گروه مالک و عموم کاربران فقط اجازه خواندن صادر شده است (r--r--)

بگونه ای که اشاره شد مجوز های دسترسی به یک فایل فقط توسط مالک آن قابل تغییر و تنظیم است.برای این کار باید از فرمان chmode استفاده کرد. برای استفاده از این دستور کاربر باید ابتدا مجوزهای دسترسی را به عددی در مبنای هشت (Octal) تبدیل نماید. مجوزهای دسترسی جمعا نه بیت است و معادل سه رقم در مبنای هشت می باشد.روش تبدیل این نه بیت به سه رقم مبنای هشت در زیر نشان داده شده است:



در جدول زیر حالات مختلف دسترسی به هر فایل و رقم معادل مبنای هشت آن فهرست شده است:

r	W	X	معادل مبنای هشت	معادل مجوز
0	0	0	0	---
0	0	1	1	--x
0	1	0	2	-w-
0	1	1	3	-wx
1	0	0	4	r--
1	0	1	5	r-x
1	1	0	6	rw-
1	1	1	7	rwx

فرض کنید فایلی با نام foo داشته باشید وبخواهید مجوز های دسترسی به آنرا بصورت زیر تنظیم کنید:

- به عنوان مالک, دسترسی کامل به فایل داشته باشید. (خواندن / نوشتن / اجرا)
- اعضای گروه مالک فایل فقط بتوانند آنرا بخوانند.(مثلا برای کپی کردن آن)
- عموم کاربران فقط اجازه ی اجرای آن را داشته باشند.

در این حالت مجوز های فایل باید به صورت 'rwxr----x' تنظیم شودکه معادل عدد باینری 111100001 است.شما ابتدا این نه بیت را به مبنای هشت برده و عدد ۷۴۱ را به دست می آورید.سپس از فرمان chmode بصورت زیر استفاده می نمایید:

$ Chmode 741 foo

پس از جرای فرمان فوق مجوز دسترسی به فایل foo مطابق نظر شما تنظیم خواهد شد.
البته ممکن است استفاده از chmod کمی مشکل به نظر برسد ولی برای کاربران مجرب بسیار ساده است.در بسیاری از گونه های یونیکس برای رفاه حال کاربران معمولی اجازه داده شده که در جلوی فرمان chmod مستقیما الگوی r,w,x را وارد نمایند ولی این قابلیت برای chmodعمومیت نداشته و باید مطمئن باشید که در یونیکس شما, از چنین حالتی حمایت می شود.

برنامه های SetUID

گاهی از اوقات کاربران یا پروسه ها مجبور می شوند به فایل هایی دسترسی داشته باشند که مجوز آنرا ندارند. بعنوان مثال در نظر بگیرید که یک کاربر معمولی تمایل دارد کلمه عبور خود را تغییر بدهد; لذا او بایدد رکورد مربوط به حساب کاربری خود را که در یکی از فایل های /etc/shadow و /etc/passwd ذخیره شده است, دستکاری کند.دسترسی به این فایل فقط برای یک کاربر در عالیترین سطح(root) امکان پذیر است و بقیه کاربران حق دسترسی به آن را ندارند. از طرفی نمی توان انتظار داشت که کاربران مختلف برای تغییر کلمه عبور خود مزاحم مسئول سیستم (administrator) بشوند و از او بخواهند که کلمه عبور آنها را عوض کند☺

از این گونه عملیات که یک پروسه با سطح مجوز پایین مجبور است به فایل هایی با سطح مجوز بالا دسترسی داشته باشد زیاد وجود دارد.در یونیکس برای آنکه چنین عملیاتی امکان پذیر باشد و در عین حال امنیت سیستم حفظ شود قابلیتی با نام SetUID ارئه شده است.
(Set User Id) با این قابلیت برخی از برنامه های خاص مجازند که وقتی توسط یک کاربر با سطح مجوز پایین اجرا می شوند, با مجوز مالک آن به سیستم فایل دسترسی داشته باشد.
بخاطر داشته باشید که وقتی کاربری یک پروسه را آغاز می نماید, سطح دسترسی آن پروسه به سیستم فایل مطابق با سطح دسترسی آن کاربر خواهد بود.برنامه های SetUID برنامه هایی هستند که وقتی توسط یک کاربر اجرا می شوند, سطح دسترسی آن برنامه معادل با سطح دسترسی مالک آن است!
(SetUID)=>Set User ID)

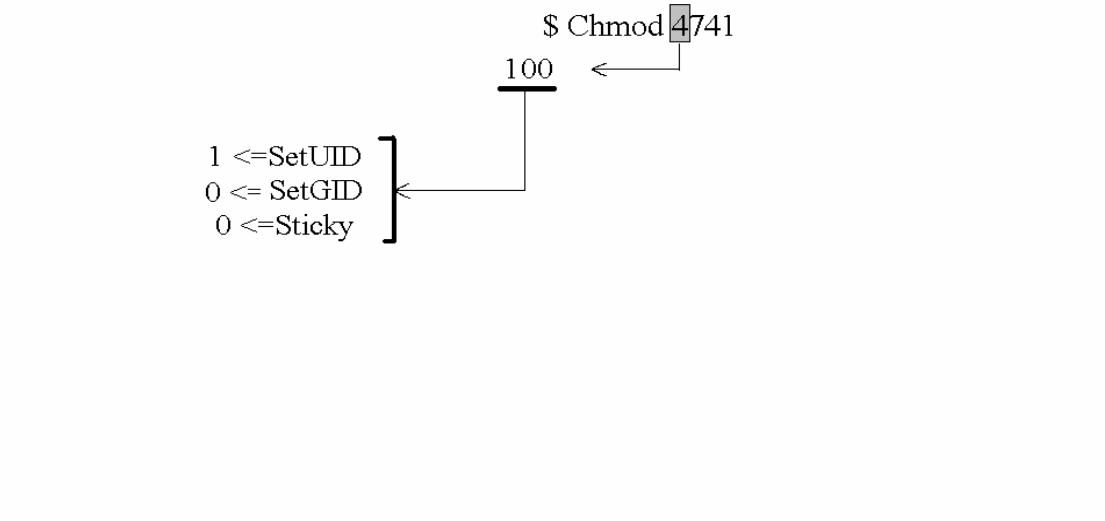
باز هم به مثال تغییر یک کلمه عبور توسط یک کاربر معمولی بر می گردیم.در یونیکس یک برنامه از نوع SetUID وجود دارد که "Passwd" نامیده می شود.مالک این برنامه root است یعنی کاربری با عالیترین مجوز دسترسی به سیستم فایل, لذا وقتی توسط یک کاربر معمولی اجرا می شود مجوز دسترسی خود را در سطح root نگه می دارد و باز هم قادر است به تمام فایل های سیستم دسترسی داشته باشد. یعنی فارغ از آنکه کسی آنرا اجرا کرده است, با مجوز سطح root اجرا خواهند شد! ولی این موضوع چیزی نیست که امنیت سیستم را به خطر بیندازد چرا که برنامه "passwd " ارتباط بسیار محدود و سختگیرانه ای با کاربر دارد. این برنامه از کاربر کلمه عبور جدید او را طلب می کند, صحت آنرا بررسی می نماید و سپس بدقت رکورد مربوطه به آن کاربر را در فایل /etc/passwd یا /etc/shadow اصلاح و باز نویسی می کند.نهایتا برنامه "passwd" خاتمه یافته و لذا کاربر هیچ مجوزی بالاتر از آنچه که داشته نخواهد یافت.
برنامه هایی که قابلیت SetUID دارند به کاربر اجازه دسترسی موقت و کنترل شده به فایل های با سطح مجوز بالا را می دهند تا عملیات مورد نیاز انجام شود.
برنامه های SetUID به غیر از ۹ بیت مربوط به تنظیمات مجوز دسترسی ,سه بیت اضافه تر برای تنظیم قابلیت اجرائی SetUID دارند.این سه بیت که قبل از ۹ بیت تنظیمات مجوز دسترسی قرار می گیرند عبارتند از:

- **SetUID bit:** مشخص کننده آنست که اگر برنامه توسط کاربر اجرا شود, مجوز دسترسی آن برنامه در سطح مجوز مالک, باقی خواهد ماند.

- **SetGID bit:** مشخص کننده آنست که اگر برنامه توسط یک گروه اجرا شود, مجوز دسترسی آن برنامه در سطح مجوز مالک گروه مالک باقی خواهند ماند.

- **Sticky bit:** برنامه مربوطه را وادار می کند تا بطوردائم در حافظه اصلی باقی بماند.

برای تنظیم این سه بیت نیز مشابه با نه بیت قبلی می توان از فرمان chmod استفاده کرد, البته به شرطی که این سه بیت نیز در مبنای هشت بیان شوند. بعنوان مثال اگر بخواهیم زمانی که فایل اجرائی foo توسط کاربران معمولی اجرا می شود, سطح دسترسی root داشته باشد باید از فرمان chomod بصورت زیر بهره بگیریم:



(البته اعمال این مجوز به شرطی امکان پذیر است که فایل foo خود در سطح root یعنی مسئول سیستم باشد) عدد ۷۴۱ بدین معناست که بیت SetUID فعال شده در حالی که بیتهای SstGID و Sticky غیر فعالند.وقتی از فرمان ls –l استفاده می کنید اگر برای فایلی بیت SetUID تنظیم شده باشد بجای گزینه x (قابلیت اجرا) گزینه S درج می شود.مثال زیر این موضوع را نشان می دهد:

$ ls –l /usr/bin/passwd
-r-s--x--x 1 root root 12244 Feb 7 2000 /usr/bin/passwd

برنامه هایی که گزینه SetUID آنها فعال است اندکی از دیدگاه امنیتی خطرناک هستند,بالاخص اگر مالک آنها در سطح مجوز root باشد. بگونه ای که گفته شد این برنامه ها فعل و انفعال بسیار محدودوتحت کنترلی یا کاربر دارند پس مشکل نا امنی آنها در کجاست؟مشکل از آنجا ناشی می شود که اگر کاربری بتواند کنترل برنامه ها را از روند عادی خارج کرده و در دست خود بگیرد, درعالیترین سطح به منابع سیستم دسترسی خواهد داشت. یکی از روش هایی که ممکن است بتوان بر اساس آن کنترل یک پروسه را بدست گرفت روش سرریز کردن پشته (Stack Overflow) است.لذا برنامه های نوع SetUID باید با دقت بسیار زیاد نوشته بشوند تا اولا قابل نفوذ نباشند و ثانیا حداقل ارتباط وفعل و انفعال را با کاربر داشته باشند ثالثا دسترسی به منابع سیستمی سطح بالا بشدت کنترل شده انجام شود.
دقت کنید که یک مسئول سیستم قادر است برنامه های زیادی را در حالت SetUID تنظیم نماید, لذا باید به دقت فهرست این برنامه ها را در خاطر داشته باشد چرا که وجود هر برنامه با قابلیت SetUID روی سیستم می تواند علامت بروز یک خطر تلقی شود. برای آنکه فهرست تمام برنامه ها با قابلیت SetUID را بدست بیاورید می توانید از فرمان زیر در محیط یونیکس بهره بگیرید:

$ find / -user root –perm -4000 –print

گزینه های مختلف فرمان فوق به شرح زیر است:

- **Find:** نام برنامه ای اجرائی در یونیکس که یک فایل با مشخصات خاص را جستجو می کند.
- **/:** جستجو باید از شاخه ریشه (root) شروع شود.
- **Root:** باید فایل هایی پیدا شوند که مالک آنها دارای مجوز سطح root است
- **perm–:** فایل هایی مورد نظر هستند که بیت SetUIDj
- **print–:** نتیجه جستجو باید روی صفحه نمایش ظاهر شود.

****

امیدوارم مفید بوده باشه!!!
گروه امنیتی آشیانه

WWW.Ashiyane.Net **&** **WWW.Ashiyane.Com**

تهیه شده توسط zeus