Как се пише курсова работа за курса по мрежова сигурност (и не само)

Васил Колев
<vasil@ludost.net>

Юни 2004
София

Съдържание

След една година водене на курса по мрежова сигурност, и проверка на курсови работи на студенти, стигнах до извода, че едно от нещата, които не се преподават, и не е ясно на студентите, е как точно се оформя и пише такъв документ. Проблемите варират от лошо оформление, през правописни и стилистични грешки, до фактологични и логически такива. Целта на този документ е да обясни тези проблеми и избягването им. За улеснение, самият той е написан в такъв вид.

Избягвал съм да давам конкретни примери от курсови работи.

Основни моменти по съдържанието на темите

Първата част на този документ се занимава със съдържанието на самата курсова работа. Това са най-сериозните проблеми, и за това смятам да обърна внимание първо на тях.

Езикови проблеми

Първият и много сериозен проблем, който прави много курсови работи неприятни и трудни за четене, е голямото количество правописни, пунктуационни и стилистични грешки. За първите извинение няма - spell checker се инсталира елементарно за каквато и да било работна среда[1] (авторът няма никакъв проблем под debian gnu/linux, както ще се види по-надолу в документа). Много често срещани са пунктуационните грешки, най-вече липсата на запетайки, или поставянето им на грешно място[2].

Стилистичните грешки са от типа да се използват грешни времена, неправилна подредба на

1 както ще се види по-надолу в документа, това не беше проблем и под Debian GNU/Linux

2 Тези два вида грешки по принцип не би трябвало да се проявяват при хора, имащи завършено основно образование в българско училище.

изреченията, или много повторения. Последното може да бъде избегнато чрез пренаписване на изреченията, и правилното оформяне на потока на мисълта. Българският език има достатъчно средства за решаването на подобни проблеми.

Ще дам следните примери за текст със стилистични грешки, и такъв, в който са поправени[3]:

През 1988 Робърт Морис написа първия Internet worm (червей). Този червей заразявал машини през няколко различни уязвимости, основно базирани на stack overflow. Така червеят се яви една от първите сериозни заплахи за мрежата.

Първият Inetrnet worm (червей) е написан от Робърт Морис през 1988. Вектор на разпространението му са били няколко различни уязвимости, основно stack oveflows. Това представлява една от първите сериозни заплахи за мрежата.

Терминология

Много често няма яснота кои термини трябва да се преведат, и кои не. Някои хора се спират на третия вариант - да напишат термините на кирилица, както се произнасят на български[4] . Ако терминът, който искате да използвате, няма известен на вас вариант на български (или не такъв, който ви се иска да използвате - като например ЗУТМД[5]), използвайте английският вариант, като използвате българските окончания за множествено число при нужда.

Примери:
page (страница от паметта) си е все страница, множественото число е pages или page-ове.
vulnerability се превежда уязвимост.
debuger си се използва по същия начин.
memory allocation (заделяне на памет) си остава заделяне на памет, не алокиране на памет.
parent process е добре да се използва пак по същия начин[6].
search engine - някой би го превел 'търсещ двигател', но на български се е наложил термина 'търсачка'.

Източници на информация

Много важна част от една тема са източниците на информация, и тяхното използване. Основните грешки, които се допускат, са лошия подбор на източници, използването на непроверена информация и разчитането на един източник, преписването направо от някои източници, както и вмъкването на ненужна информация.

Лошият подбор на източниците се състои в това, че се вземат рекламни брошури като източници на информация, и се набляга на минимизация на броя източници на информация. Така се получава повтаряне на чужди грешки и разпространението им. Малко са източниците, в които има перфектна информация, а съществуването на глобална мрежа, от която можем да черпим информация, и локална човешка такава не оставят никакво извинение за ограничаването на информацията, която използваме.

Основният проблем - използването на рекламни брошури - е лесен за преодоляване, просто като се обърне внимание на представянето на информацията. Ще дам за пример следната такава брошура:

---

3 И двата текста не претендират за сериозна информационна стойност.

4 Това е и най-грешния вариант, например интеджер вместо integer

5 ЗУТМД е записващо устройство с твърд магнитен диск (т.е. harddisk)

6 може и като parent процес, но 'процес-родител' звучи странно на български

> GreatProduct 2001, следващата версия на продукта след GreatProduct 1000, е крайъгълния камък на GreatProduct System. GreatProduct 2001 помага за по-ефективната работа като създава условия потребители, информация и фирмени процеси да бъдат свързани помежду си с помощта на GreatCorp технологии за съвместна работа, Information Rights Management (IRM), поддръжка на Extensible Markup Language (XML) и др. В различните версии на GreatProduct 2001 са включени следните програми:
> ...

Стилът на по-горния текст е лесно различим - в него не се дава някаква сериозна информация, споменават се само 2 (нови) технологии, за да изглежда добро и модерно, и останалото са хубави думички, които нищо не казват за възможностите на продукта. Да видим нещо друго:

> Productos е клонинг на операционната система LazyAuthorix, написана от нулата от Пешо Кернелски с помощта на група програмисти, разпръснати из Internet. Тя цели POSIX и Single Unix Specification съвместимост.
>
> В нея има повечето възможности, които бихте очаквали в модерна LazyAuthorix система, включващи истинска многозадачност, виртуална памет, shared libraries, demand loading, shared copy-on-write изпълними файлове, правилно управление на паметта, и TCP/IP мрежова поддръжка.

За разлика от предишния текст, този споменава спокойно основните възможности на продукта, накратко дава историята му, и прави нормално сравнение с други подобни продукти. Също така дава две различни спецификации, които описват системи като нея[7].

На практика, и двата текста не са много полезни при разработката на курсова работа, за това нека да видим един трети [RFC2821]:

> An SMTP reply is an acknowledgment (positive or negative) sent from receiver to sender via the transmission channel in response to a command. The general form of a reply is a numeric completion code (indicating failure or success) usually followed by a text string. The codes are for use by programs and the text is usually intended for human users. Recent work [34] has specified further structuring of the reply strings, including the use of supplemental and more specific completion codes.

Ето това вече е пример за точна и ясна техническа документация, която би била полезна.

Друго важно за източниците е, че рядко можем да разчитаме само на един. Много полезно за разработващите такива теми е да определят един основен източник на информация за даден продукт, и поне един алтернативен, така че да могат сами да преценят истинността на дадена информация. Сравняването и проверката на информацията е един от основните етапи в писането на курсова работа.

Преценяването на основният източник рядко е трудно - например всеки свободен софтуер си има някакъв основен форум (mailing листа) като основен източник на информация (някои имат и

---

7 Понякога е трудно да се направи разлика между маркетинговите термини и чисто техническите. В такива случаи е добра идея да се консултирате с някой, който работи в областта.

резюмета на основния източник, като например [KT] и [LWN] за [LKML]). Съответно комерсиалните продукти имат страници на производителите[8], и понякога публични (и поддържани от производителя) форуми.

Подходящите алтернативни източници най-често се поддържат и използват от потребители на дадения продукт, и не е трудно да бъдат открити чрез различни търсачки. Те дават един допълнителен поглед върху продукта и ни допълнителна информация за продукта, която може да липсва в основния източник (или просто да е различна[9]). Също такива полезни източници са различни статии и рецензии на продукта.

Има и трети вид източници на информация - насочващи, които много често се бъркат с основните. Това са например различните сайтове, които правят бази данни с security проблеми. Те са полезни за намиране на посока за по-нататъшно търсене, и за изясняване на някои ключови моменти, но не и като единствен източник, в който има цялата нужна информация.

Това разделение на източниците важи и за неща, които не са точно продукти - нека дам един пример с buffer overflows. Основен източник би ни била статията на Aleph One, "Smashing the stack for fun and profit". След това, като разберем основните принципи, можем да потърсим в сайта на основния източник (в нашия случай – [phrack]) допълнителна информация и други статии по въпроса, след което, събрали тази информация, можем да намерим алтернативни източници чрез google, като търсим специфични неща при heap overflows, off-by-one атаки и т.н. Можем да използваме сайт като securityfocus, за да съберем информация за честотата на използване на тези атаки, и за отправна точка към различни хора и компании, които се занимават с намиране и решаване на такива проблеми (или с писане на exploit-и).

Проблем се получава и от това, че поради недостатъчно източници в темата се пропускат важни неща, които не са описани навсякъде. Много често, поради четене на рекламни брошури се пропускат други продукти със същата функционалност, други възможности на системите (и например пробивите в тях), и т.н. Много важно за една такава курсова работа е да е изчерпателна, да покрива поне основните неща по темата.

Следващия проблем с източниците и грешния им подбор е вмъкването на ненужна информация. Случва се доста често, поради неразбиране на материала, да се вмъкне някакъв параграф, който изглежда на място, и всъщност няма нищо общо с темата, или да се вмъкне материал, който има отношение към темата, но няма смисъл от него. Например, в тема, който разглежда сигурността на FreeBSD ядрото няма смисъл да се слага source кода на цялото ядро, въпреки, че той има връзка с темата.

Минимализъм и елегантност

Този проблем е тясно свързан с нещо, към което трябва винаги да се стремим - минимализъм и елегантност[10]. Много подходящ е един цитат на покойния Edsger W. Dijkstra, "*...elegance is not a dispensable luxury, but a factor that often decides between success and failure*." [EWD1273]. Целта на една тема като цяло е да е използваема, т.е. в нея да има нужната информация, без неща, които да ни отклоняват или пречат да я намерим и използваме. Затова например уводите трябва да са кратки и стегнати, да се въздържаме от ненужни отклонения, и от вмъкване на безсмислена информация и усложнени изрази, така че да правим четенето трудно.

---

8 факт е, че много такива сайтове затрудняват достъпа до полезната техническа информация, или просто я нямат.

9 добра презумпция е, че фирмите лъжат, до доказване на противното.

10 не само в курсовите си работи

За да покажа по-добре идеята за минимизация, ще дам следния пример - нека трябва да пишем курсова работа, в която се обсъждат syscall-овете, използвани при работа с бази данни под Linux:

> В операционната система Linux (чиито представители са RedHat Linux, SuSe linux, Debian GNU/Linux) системното извикване (syscall) open (което няма wrapper в libc6) може да се зададе флаг O_SYNC (чрез побитови операции с втория параметър). Този битов флаг ни дава синхронна работа с устройството, на което се намира файла, и е полезно например за бази данни. За пръв път се появява в SGI Irix (където има и fcntl операция върху descriptor, която може да доведе до същия ефект).

> open приема флаг O_SYNC, който води до синхронна работа с файла, т.е. приложението може да е сигурно, че данните му са достигнали физическия носител след връщането на функцията.

Вторият текст в контекста на темата съдържа нужната информация - няма смисъл да обясняваме, че сме под linux, кои са му представителите (което е и неточно като термин), откъде произхожда, и че е полезно за бази данни - нали в крайна сметка това е темата.

И като последно, изискването за обем - много хора се притесняват от него, и за това вмъкват много ненужна информация (на което се спрях по-горе). Понеже проверяващите всъщност наистина четат курсовите работи[11], качеството се цени много повече, отколкото количеството, и дори има шанс една разводнена тема, която обаче отговаря на изискването за обем, да получи по-ниска оценка от такава, която не е достатъчно голяма, но съдържа нужната информация в стегнат вид.

Разликите между цитиране и преписване

Една от причините да се появяват доста от по-горе описаните проблеми е, че не се прави разлика между цитиране и преписване, и изобщо се преписва/превежда безогледно. Целта на една тема не е да се покаже колко добре може да се налепят парчета информация, намерени от няколко места, а да се напише от студента, и да се покаже, че той разбира за какво става въпрос, и може да го обясни със свои думи. В крайна сметка, обработката и разбирането на много информация са много важни за всеки, който работи в областта на сигурността (и в информатиката като цяло).

В тази връзка препоръчвам на всички да прочетат статиите на Edsger W. Dijkstra на тема академичното обучение. Една от тях е спомената в библиографията - [EWD1252].

Външно оформление

Всяка курсова работа се чете поне по няколко пъти - и от преподавателите, и от хората, които после свалят публикувания вариант през Internet. По тази причина оформлението трябва да се направи по начин, който да не дразни четящите, и е удобен за преглеждане и четене.

Много важен момент за всички теми е, че трябва да са приятни за четене на хартия, и в оформлението им да се изхожда от тази презумпция.

Първият момент е, че електронният формат на предадената тема трябва да е PDF. Този формат е избран, защото се поддържа от всички операционни системи, има много публични

11 колкото и невероятно да се вижда това на студентите

инструменти за работа с него и конвертиране към него, и видът му на екрана и на хартия е един и същ (което се дължи на стабилната му спецификация).

Използваните шрифтове трябва да са SansSerif-ни (TimesNewRoman е Serif-ен шрифт, Arial, FreeSans са SansSerif), поради това, че по-лесно се четат. За source code, имена на функции, вход/изход на машина и т.н., е важно да се използва `fixed-width` шрифт, а не пропорционален (подходящ такъв шрифт е `monotype` или варианта му `FreeMono`). Специално цитирането на source код би трябвало да изглежда така[12]:

src/linux-2.6.7/net/ipv4/ip_options.c:

```
87          struct ip_options *sopt;
88          unsigned char *sptr, *dptr;
89          int soffset, doffset;
90          int     optlen;
91          u32     daddr;
92
93          memset(dopt, 0, sizeof(struct ip_options));
94
95          dopt->is_data = 1;
96
97          sopt = &(IPCB(skb)->opt);
98
99          if (sopt->optlen == 0) {
100                 dopt->optlen = 0;
101                 return 0;
102         }
103
104         sptr = skb->nh.raw;
105         dptr = dopt->__data;
```

Така става веднага ясно откъде е source, и за кое в него се говори - например, че на ред 98 се занулява структурата с опциите, и това лесно може да се намери и в оригиналния файл.

По същия начин би трябвало да се показва работа с терминал:

```
vasil@marla:~$ id                    проверяваме кой потребител сме
uid=1000(vasil) gid=1000(vasil) groups=1000(vasil),4(adm),30(dip),1012(chise)
vasil@marla:~$ pwd                   проверяваме текущата директория
/home/vasil
vasil@marla:~$ uptime                проверяваме натоварването на машината
 13:00:51 up 67 days, 52 min,  3 users,  load average: 3.14, 2.01, 1.87
```

Нормалния изход от терминала е с нормален стил, с `fixed-width` шрифт, нещата, които пише потребителя, са **bold,** а коментарите са *italic,* с 1pt по-малък размер и с нормалния шрифт.

Повечето софтуер за текстообработка поддържа възможности като бележки под линия, които са много удобни за работа и помагат да не се отклонява вниманието на читателя от основния текст. Повечето от хората, работещи в областта на информатиката нямат проблем да пишат многопоточно, и

12 Подобен стил може да се срещне в много книги, като например [UNP]

да вмъкват по няколко нива на скоби, но така текстът става малко по-труден за четене, и за това се препоръчва когато е възможно, да се избягват подобни усложнения.

За съжаление, софтуерът много успешно може да ни помогне да се простреляме в крака – например да не съобразим, че показва хипервръзките като някакво подчертано заглавие, и след като щракнем на него, и казва накъде сочат. Това е прекрасна идея за електронни документи, но не се пренася подходящо на хартия. Препоръчвам да следва стила на този документ, където повечето връзки са описани накрая в библиографията, и в текста се споменават със съкращението, което им е заглавие. Например връзка към този документ би била да се спомене, че един от малкото известни ми документи, които правят рекурсивна връзка към себе си, е [BK1].

Как беше написан този документ

По-голямата част от материала беше написана в GEdit, и проверена с aspell за правописни и печатни грешки. След това текстът беше прехвърлен в OpenOffice 1.1.1, и оформен с шрифтовете FreeSans и FreeMono[13]. Резултатът беше конвертиран в PDF чрез функция на OpenOffice, и резултатът беше допълнително прегледан с xpdf. За финалните прегледи беше печатан през CUPS, на стар принтер LaserJet4, на нормална хартия.

Информация за подходящи инструменти за работа под други операционни системи може да се намери [NSEC-Forum].

13 които могат да бъдат открити в пакета ttf-freefont в Debian, и се разпространяват с много FOSS програми и пакети

Приложение 1 – Примерен план за разработка на курсова работа

Както всички планове, които давам, и този е препоръчителен. Може да го следвате, може да го модифицирате, а желаещите могат да си кажат, че е глупав, и да измислят собствена методика за работа .

Този план лесно се модифицира за почти всичко, както и за разработка на софтуер, така и за доказване на теорема. В него няма нищо ново или гениално, и представлява приблизителна форма на моя начин на писане на подобни разработки.

1. Сформиране на екипа, който ще я пише/разработва
2. Изясняване на целта и темата
3. Намиране и сортиране на източниците на информация
4. Събиране на нужната информация
5. Анализ на информацията
6. Оформяне на първа версия
7. Преглед, допълване на липсващи елементи
8. Няколко прегледа на получената тема, за изчистване на грешки
9. Оформяне на финалния документ
10. Няколко препрочита от различни хора, за откриване на последните грешки

Приложение 2 – Библиография

Примери за източници на информация бяха вземани от:

http://www.microsoft.com/bulgaria/office/office2003/applications.mspx
http://www.kernel.org/
[RFC2821] RFC 2821, Simple Mail Transfer Protocol
ftp://ftp.ietf.org/rfc/rfc2821.txt
[phrack] Phrack Magazine
http://www.phrack.org/
[KT] Kernel Traffic,

http://www.kerneltraffic.org/
[LWN] Linux Weekly News
http://lwn.net/
[LKML] Linux-Kernel Mailing List, FAQ може да се намери на адрес
http://www.tux.org/lkml/

Идеи за разботката са използвани от:

[EWD1252] Edsger W. Dijkstra, “Elegance and efective reasoning”,
http://www.cs.utexas.edu/users/EWD/ewd12xx/EWD1237.PDF
[EWD1252] Edsger W.Dijkstra, “Convocation speech”,
http://www.cs.utexas.edu/users/EWD/ewd12xx/EWD1252.PDF
Умберто Еко, “Как се пише дипломна работа”, ISBN 954-455-022-5
[UNP] W.Richard Stevens, “Unix Network Programming vol.1”, ISBN 0-13-490012-X
[BK1] Васил Колев, “Как се пише курсова работа за курса по мрежова сигурност”,
http://vasil.ludost.net/pisaniq/howto-cp.pdf

Други:

[NSEC-Forum] Форум на курса по Мрежова сигурност към ФМИ на СУ
http://nedyalkov.com/forum/list.php?f=3

Проблемите, описани в документа, са наблюдавани в курсовите работи на студентите от курса по по мрежова сигурност във Факултета по Математика и Информатика към СУ “Св. Климент Охридски”, http://nedyalkov.com/security/