техника переполнения кучи в xBSD

крис касперски, ака мыщъх, no-email

атаки на кучу (она же динамическая память) приобретают все большую популярность, но методы переполнения, описанных в доступных хакерских руководствах, по ходу дела оказываются совершенно неработоспособными. рассчитанные на древние системы, они не в курсе, что в новых версиях все изменилось. более того, Free-, Net- и Open-BSD используют _различные_ стратегии защиты кучи, особенности строения которой необходимо учитывать при написании exploit'ов. мыщъх проделал титаническую работу, исследовав _все_ версии _всех_ BSD-систем и теперь не успокоится пока не поделится добытой информации с читателями

введение или как это все начиналось

Сезон переполняющихся куч начался со статьи "Once upon a free()", опубликованной анонимусом в 39h номере phrack'a (8 января 2001 года) со ссылкой , ~~ссылающегося~~ на более раннюю работу известного хакера Solar'a Designer'a (датируемую 25 июлем 2000 года), в которой тот описал механизм передачи управления на shell-код, использующий особенности реализации макроса unlink в библиотеке glibc-2.2.3, поставляемой вместе с Linux. До этого, хакерам было известно лишь стековое переполнение с подменой адреса возврата из функции. Переполнение кучи в общем случае приводило лишь к бездумному разрушению служебных структур динамической памяти и краху уязвимого приложения. Новый класс атак открывал нехилые перспективы непаханой целины и обозначенная статья приобрела огромную популярность, породив кучу перепечаток в различных туторалах и емагах.

В конце июля 2002 года на сеть обрушится червь Slapper, поражающий Linux-боксы и распространяющийся путем переполнения кучи через ошибку переполнения стека в OpenSSL, что представляло собой весьма нетривиальную задачу.

Для BSD-систем такой механизм оказался неприменим в силу существенных конструктивных отличий строения кучи, но уже 14 мая 2003 года хакер по кличке BBP обосновал как атаковать аллокатор, разработанный программистом по имени Poul Henning Kamp (далее по тексту phk-аллокатор). В то время phk-аллокатор использовался в Free-, Net- и OpenBSD в качестве основного аллкатора, выбираемого по умолчанию, теперь же он остался только в Net- и OpenBSD (причем, в OpenBSD – в сильно переработанном виде), в результате чего старые трюки перестали работать и молодые хакеры, обкурившиеся чужих статей, никакого кайфа не словили, а высели на полный облом, завалив мыщъх'а грудами писем почему это не работает, как теперь жить и что делать?

Вердикт: курить, конечно, полезно, но реальный приход дают только свои собственные исследования, особенно в наш бурный век, когда опубликованные методы атаки теряют актуальность в течении нескольких месяцев. Так вот, на счет исследований.

В конце 2005 году хакер по кличке Phantasmal Phantasmagoria написал статью "The Malloc Maleficarum Glibc Malloc Exploitation Techniques", предлагающую новые механизмы атаки, пробивающие glibc версии 2.3.5 и выше. Статья остается актуальной и по сей день, пускай, и не без коррекции с учетом рандомизации адресного пространства, сторожевых страниц и других новомодных защитных технологий, появившихся в начале 2006 года.

В середине 2006 года хакер по кличке Ben Hawkes выступил на конференции RexCon с презентацией "Exploiting OpenBSD" в которой он показал как атаковать кучу самой защищенной операционной системы всех времен и народов. Разработчики OpenBSD довольно оперативно отреагировали на ситуацию, впустив обновленную версию аллокатора, затыкающую _часть_ дыр. Разработчики Free- и NetBSD чешутся до сих пор, так что для хакерства складывается весьма благоприятная ситуация.

Мыщъх не предлагает готовых рецептов, но зато указывает на направление, двигаясь в котором можно подломать не только текущие, но и последующие версии аллокаторов во Free-, Net- и OpenBSD-системах, включая Linux и ненавистную всем Вислу с новоявленным Longhorn'ом (он же Server 2008).

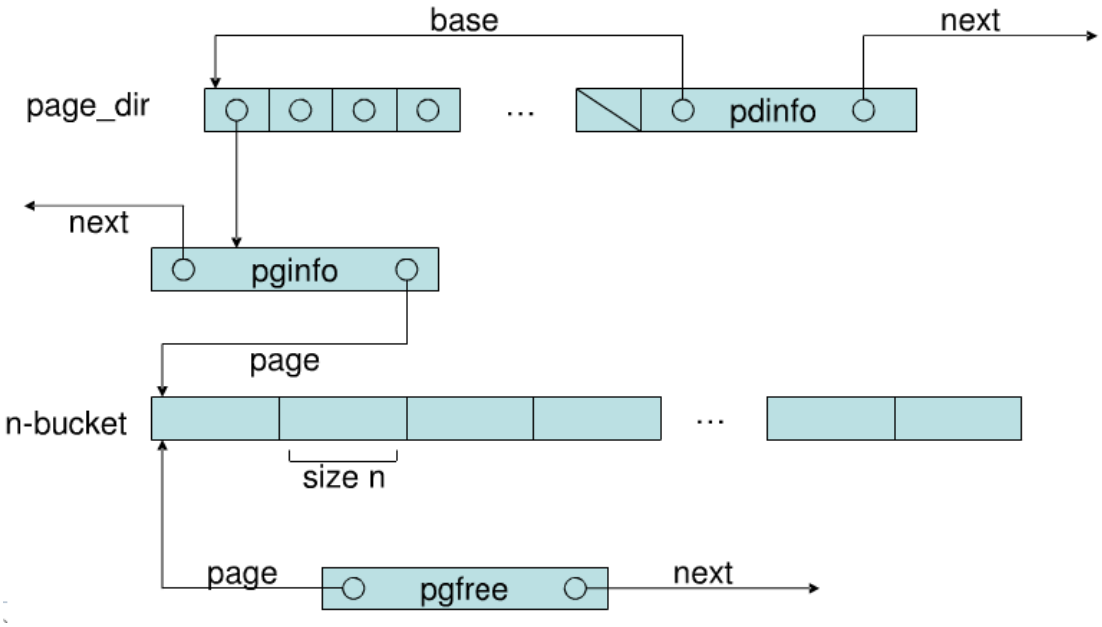

Рисунок 1 схематичное строение кучи в BSD-системах

обзор новых защитных технологий

Прежде, чем углубляться в омут технических подробностей (в который легко занырнуть, но не каждому удается вынырнуть наружу), совершим беглый экскурс по новейшим аллокаторам, обозначив ключевые технологии и ответив на вопрос почему существующие exploit'ы перестали работать.

- **изменение структур данных**:
  - heap smashed exploit'ы вынуждены работать с низкоуровневыми структурами данных, поддерживающих жизнеобеспечение кучи, причем, эти структуры не остаются постоянными, а меняются от версии к версии, не только в целях защиты, но и элементарной оптимизации. поэтому, чтобы заставить exploit работать, необходимо ~~установить~~ определить версию аллокатора, используемую жертвой, скачать исходные тексты (а в случае Windows – засесть за дизассемблер) и скорректировать код exploit'а соответствующим образом. если же версию аллокатора установить не удается (а удаленно ее ~~установить~~ определить очень сложно), следует поочередно перебирать все версии одну за другой;
- **рандомизация адресного пространства**:
  - exploit'ы прежнего поколения закладывались на абсолютные адреса, по которым были расположены ключевые структуры данных, указатели и другая информация, пригодная для затирания, теперь же затереть ее не так-то просто, поскольку операционные системы нового поколения активно используют рандомизацию адресного пространства (Address Space Layout Randomization или, сокращенно ASLR), меняя стартовые адреса библиотек и служебных структур динамической памяти случайным образом. OpenBSD и Висла используют рандомизацию кучи по умолчанию, в NetBSD она до сих пор не реализована, а разработчики FreeBSD в последних версиях отказались от ее использования в угоду производительности, так что реальную угрозу для хакерства рандомизация представляет только на OpenBSD (рынок которой относительно невелик) и на Висле/Longhorn'е;

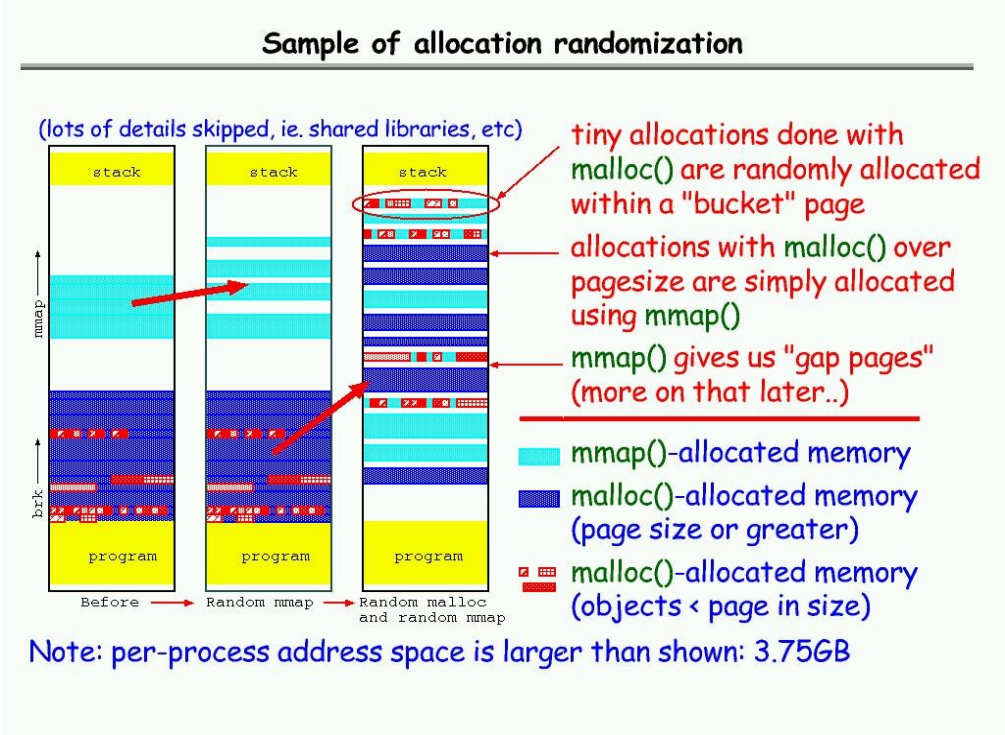

Рисунок 2 рандомизация адресного пространства в OpenBSD

- **сторожевые страницы**:
  - для предотвращения переполнения блоки памяти, занимающие свыше 4 Кбайт, окружены сторожевыми страницами, попытка доступа к которым вызывает исключение, предотвращая переполнение, однако, блоки памяти меньше 4 Кбайт остаются незащищенными (в противном случае расходы памяти оказались бы просто немыслимыми) и к тому же _внутри_ блоков память _никак_ не защищена, то есть, если мы имеем структуру вида struct X{char buf[0x10]; int *x; int *y}, то перезапись указателей x и y по прежнему остается возможной! сторожевые страницы в настоящий момент реализованы только в OpenBSD (см. http://www.openbsd.org/papers/auug04/mgp00023.html), но даже в ней они ~~(для экономии памяти)~~ могут быть выключены для экономии памяти;
- **контроль целостности кучи**:
  - практически все современные аллокаторы в той или иной мере контролируют целостность служебных структур динамической памяти, предотвращая их затирание при переполнении, однако, зная алгоритм проверки, мы можем засунуть в затертые структуры подложные данные и… никто ничего не заметит! как вариант: мы можем найти в куче указатель на блок памяти, передаваемый функции free() и заменить его указателем на свой собственный блок, содержащий поддельную структуру данных;
- **обфускация указателей**:
  - для предотвращения чтения/записи указателей прибегают к их "шифровке" командой XOR. указатели хранятся в памяти в зашифрованном виде и расшифровываются только перед непосредственным использованием, а потом зашифровываются вновь. в настоящий момент данный защитный механизм реализует только Server 2008 и новые версии компилятора VC, а разработчики xBSD отказались от обфускации указателей ввиду чрезмерных накладных расходов, хотя OpenBSD позволяет включить обуфскацию (по умолчанию она выключена);

история

Главная сложность атаки на кучу заключается в огромном количестве версий аллокаторов, каждая из которых требует индивидуального подхода, поэтому, представляет большой интерес проследить этапы развития аллокаторов во Free-, Net- и OpenBSD системах.

Полный перечень изменений занял бы пару десятков страниц и был бы таким же скучным и труднопроходимым как "Капитал" Маркса (часто используемый в качестве снотворного).

К счастью, ничего подобного делать не требуется, поскольку эта информация уже представлена в удобочитаемом виде на сайтах разработчиков. Web-CVS рулит когда требуется изучить хронологию эволюции отдельного файла — в данном случае malloc.c, доступном по следующим адресам:

- FreeBSD:
  - http://www.freebsd.org/cgi/cvsweb.cgi/src/lib/libc/stdlib/malloc.c;
- NetBSD:
  - http://cvsweb.netbsd.org/bsdweb.cgi/src/lib/libc/stdlib/malloc.c;
- OpenBSD:
  - http://www.openbsd.org/cgi-bin/cvsweb/src/lib/libc/stdlib/malloc.c;

Мы же сосредоточимся только на "судьбоносных" изменениях, связанных с введением в строй новых защитных механизмов, а модернизацию служебных структур динамической памяти оставим за бортом.

FreeBSD

Ранние версии FreeBSD использовали простой и тормозной аллокатор, написанный в 1982 году программистом по имени Chris Kingsley (kingsley@cit-20) и условно именуемый Caltech-аллокатором.

В сентябре 1995 года, Caltech-аллокатор был заменен намного более продвинутым аллакатором, написанный программистом Poul-Henning Kamp <phk@FreeBSD.ORG> (phkmalloc-аллактор или для кратности "phk"), который и стал основным аллакатором для xBSD систем на последующую пару десятков лет. Выпущенный под лицензией "THE BEER-WARE LICENSE" (бесплатно как пиво) он находит себе применение и по сей день…

Начиная с октября 1995 года в phk-аллокаторе появились "zero" и "junk" опции, позволяющие диагностировать некоторые виды переполнения кучи, впрочем, в то время кучу в BSD системах переполнять еще никто не собирался и потому эти улучшения остались незамеченными.

Декабрь 1995 года ознаменовал череду радиальных изменений, направленных главным образом, на усиление производительности и повышение эффективности использования памяти. Побочный эффект — изменение служебных структур кучи — уничтожил все существующие exploit'ы, однако, по официальным данным, ни одного работающего expliut'a, атакующего кучу на тот момент еще не существовало.

Затем последовала череда многочисленных мелких изменений, за которой незаметно прошел 2001 год, отмеченный появлением exploit'ов под Linux; 2003 год, открывший эру атак на кучу xBSD; закончился 2005 год, к концу которого хакеры справились с новыми версиями glibc, а разработчики FreeBSD все никак не реагировали на ситуацию и от атак не защищались, перекладывая эту заботу на плечи компилятора (учитывая, что большинство программ использует кучу не напрямую, а работает через glibc, позицию создателей FreeBSD можно понять).

Наконец, в январе 2006 года древний phk-аллокатор был отправлен на свалку истории, а на его место пришел новый, улучшенный jasone-аллакатор, созданный программистом по имени Jason Evans. Функции malloc(), calloc(), posix_memalign(), realloc() и free() были полностью переписаны с учетом требований защиты и масштабируемости. Ну, масштабируемость нас никак не волнует, так что сразу перейдем к защите: в jasone-аллакаторе появились сторожевые "красные зоны" (redzones), располагающиеся до и после всех блоков памяти, а так же проверки на переполнения буферов, существенно затрудняющие атаку, но…

…в марте 2006 года "красные зоны" были удалены по соображениям производительности, а в CVS-дереве появился следующий комментарий: "Remove redzones, since program buffer overruns are no longer as likely to corrupt malloc data structures" – "*удалены красные зоны, поскольку переполнение буферов ничуть не более вероятно, чем разрушение внутренних структур динамической памяти*", из чего надо полагать, что ни буфера, ни внутренние структуры не были защищены. Правда, в силу своей новизны, jasone-аллактор оказался достаточно стойким и exploit'ы, ориентированные на phk-аллокатор, с ним обломались, но! базовые принципы атаки не изменились и требовалось всего лишь адоптация exploit'ов под новые структуры данных.

На момент написания этих строк (конец 2007 года) FreeBSD продолжает использовать незащищенный jasone-аллактор, представляющий собой легкую мишень для атаки.

Рисунок 3 malloc.c на CVS-дереве FreeBSD

NetBSD

Операционная система NetBSD, "отбранченная" от FreeBSD, во многом повторила ее путь. Сначала в ней использовался Caltech-аллокатор, но в июне 1999 года он был заменен phk-аллокатором, позаимствованном из FreeBSD и портированным на все платформы, которые только поддерживает NetBSD. Изменения внутренних структур данных динамической памяти оказались несущественными и потому обе системы оказались "совместимыми" с точки зрения exploit'ов.

Очередная серия изменений произошла в мае 2001 года, когда работники NetBSD вновь позаимствовали новую версию phk-аллокатора из FreeBSD, слегка адоптировав ее в соответствии со своей философией и добавив ряд мелких проверок на переполнение кучи, которые, в общем-то, оказались современно несущественными с хакерской точки зрения.

В настоящий момент NetBSD продолжает использовать незащищенный phk-аллокатор. Рандомизация адресного пространства реализована только в стеке и секциях кода/данных, но куча остается нерандомизованной. Сторожевые страницы так же отсутствуют, как отсутствует обфускация указателей, а потому NetBSD легко атакуется древними exploit'ами. Впрочем, чтобы захватить управление, хакеру необходимо преодолеть защитный механизм PaX, интегрированный в NetBSD, подробнее о котором можно прочитать в моей статье "переполнение буфера на системах с неисполняемым стеком" (см. http://nezumi.org.ru/zq-nx.uncensored.zip), но это по любому тема совсем другого разговора, не имеющего к куче никакого отношения.

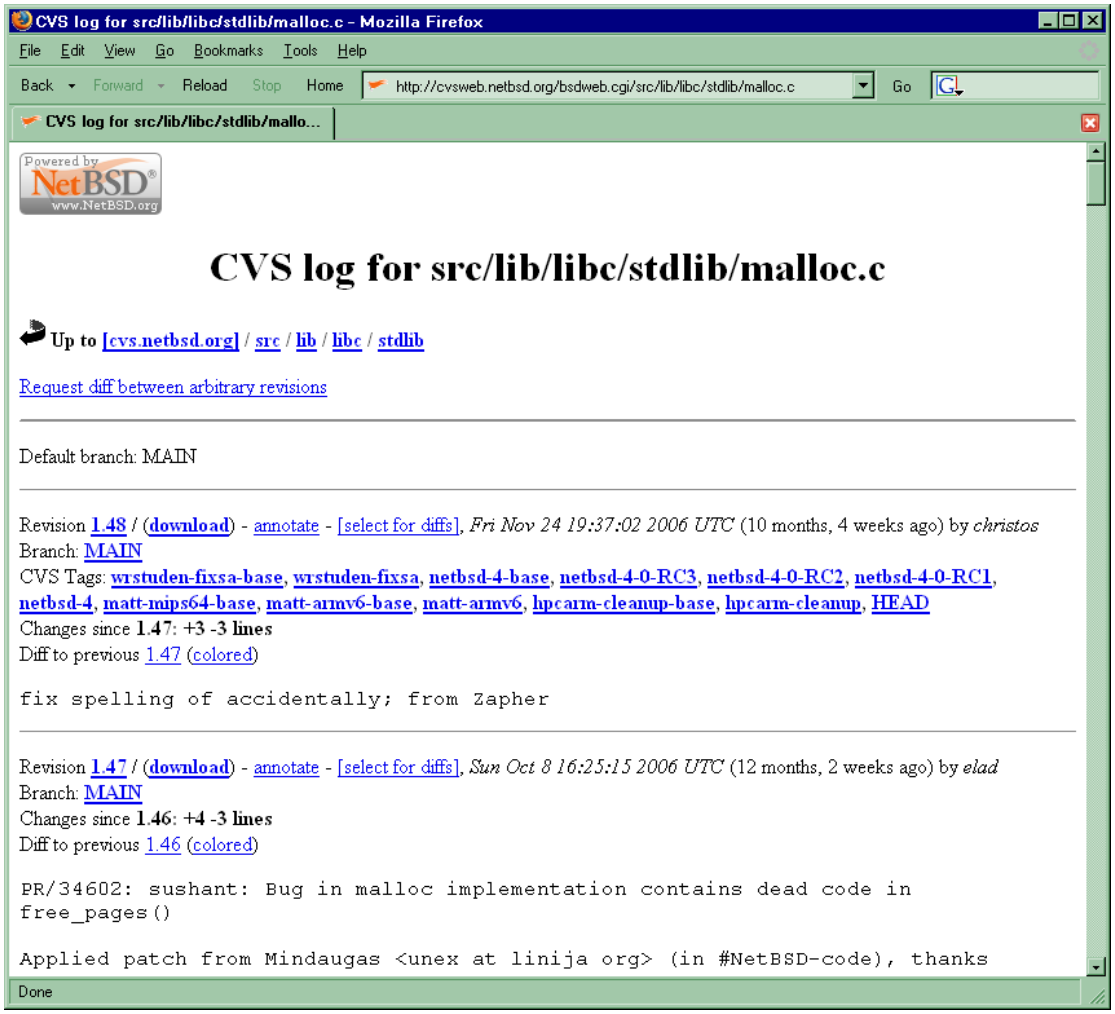

Рисунок 4 malloc.c на CVS-дереве NetBSD

OpenBSD

Ох, уж эта драконическая OpenBSD!!! Самая трудная мишень для атаки! Можно даже сказать, практически неприступная, но именно эта неприступность и возбуждает хакеров, заставляя их искать весьма нетривиальные пути. Чтобы развеять мир о неприступности OpenBSD достаточно вспомнить Ben'a Hawkes'a, атаковавшего кучу OpenBSD в 2006 году, то есть, когда в ней уже были реализованы основные защитные механизмы, отсутствующие у конкурентов, но не будем забегать вперед и вернемся назад в далекий 1995 год, когда в OpenBSD использовался Caltech-аллокатор со всеми дырами, что в нем были.

В августе 1996 года разработчики OpenBSD позаимствовали phk-аллокатор из FreeBSD без каких бы то ни было существенных переделок, что позволило хакерам атаковать OpenBSD с той же легкостью, что и остальные системы.

В октябре 2003 года (то есть, спустя полгода после открытия сезона атак на кучу, начатую хакером BBP), коллектив OpenBSD оказался первым кто отреагировал на эту угрозу и выпустил усиленную версию phk-аллокатора, добавив в него сторожевые страницы и рандомизацию кучи, что вызвало бурю восторга среди "трудящихся", которые вовсю использовали новый аллокатор на продакшен машинах, считая себя абсолютно неуязвимыми.

Коллектив же OpenBSD тем временем вместо того, чтобы почивать на лаврах, не отходил от станков, тьфу, от клавиатур, и ковал новое секретное оружие, представленное народу в августе 2004 года. Появилась обфуская указателей, а рандомизация кучи была существенно усилена и теперь она затрагивала не только адреса блоков памяти, но и местоположение служебных структур, что делало кучу полностью непредсказуемой и очень-очень сложно атакуемой, однако, в июне 2005 года по многочисленным просьбам "трудящихся" обфускация указателей была отправлена в отставку, поскольку оказалось через чур тяжеловесной и негативно влияющей на производительность.

"Наш ответ Чемберлену" — вот девиз аллокатора, выпущенного в 2006 году и устраняющего ряд слабостей реализации, продемонстрированных на презентации "Exploiting OpenBSD". Говоря техническим языков, для структур pginfo и pgfree был создан специальный аллокатор, размещающих их в отдельной области памяти (прежние версии аллокатора использовали для этих целей функцию imalloc), чанки (chunks) переместились в специальный массив, размещаемый в случайном месте адресного пространства, в результате чего затереть служебные данные кучи в последних версиях OpenBSD практически невозможно, но создать подложный чанк и "скормить" его функции free() – это без проблем! ведь обфускация указателей по умолчанию отключена. Однако, найти уязвимый указатель пригодный для затирания — удается далеко не всегда, так что OpenBSD не без ложной скромности носит звание самой защищенной операционной системы (только не надо путать "самую защищенную" с "защищенную вообще").

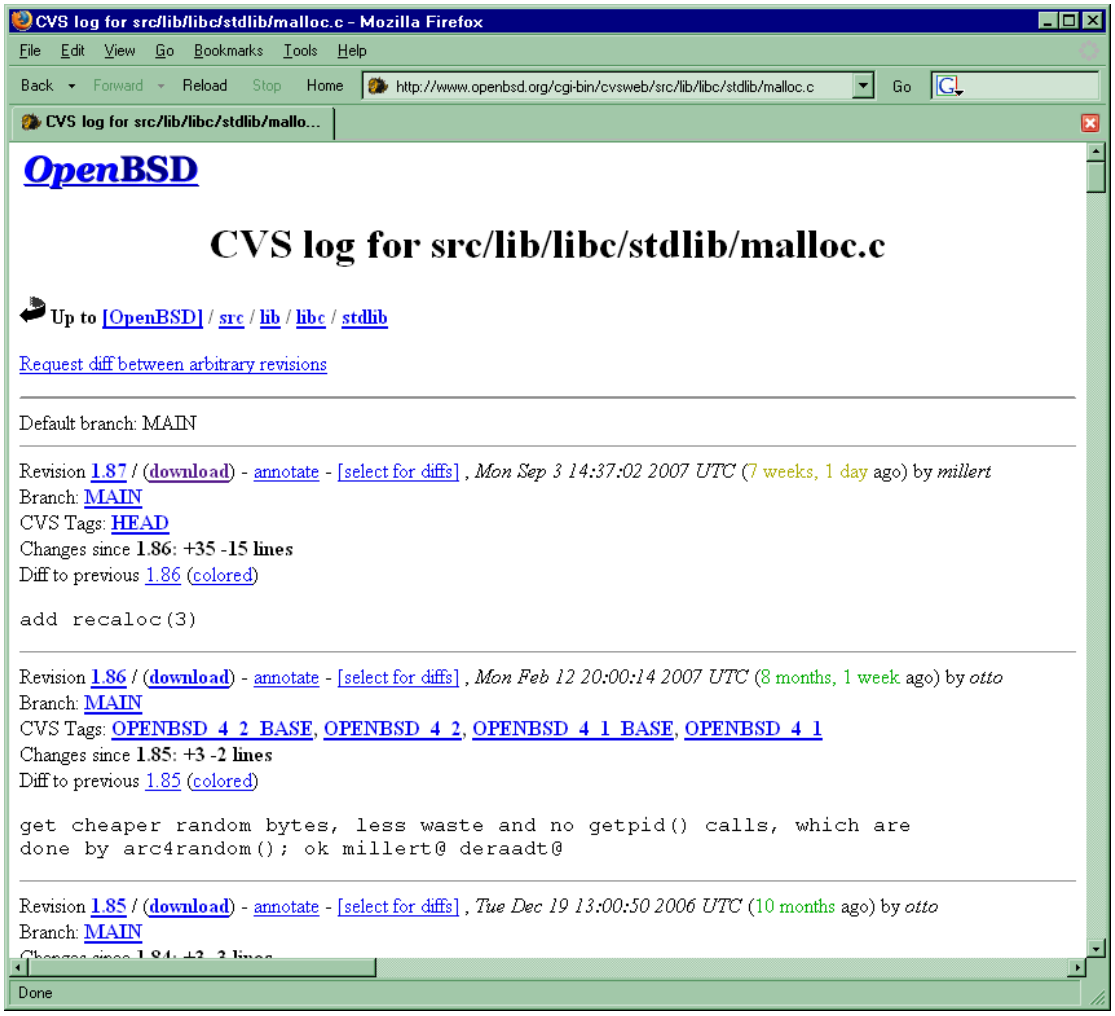

Рисунок 5 malloc.c на CVS-дереве OpenBSD

заключение

Вот мы и пробежались галопом по всем аллокаторам, которые только есть, рассмотрев их сильные и слабые стороны. И что же мы обнаружили в итоге? FreeBSD атаковать ничуть не сложнее чем NetBSD, однако, из-за различных типов применяемых в них алокаторав, exploit'ы получаются несовместимыми. OpenBSD – единственная система, существенно затрудняющая атаки на кучу, но… вовсе не делающая их принципиально невозможными!

>>> врезка ссылки по теме

- **JPEG COM Marker Processing Vulnerability in Netscape Browsers by Solar Designer:**

- описание уязвимости в браузере Netscape, обнаруженное Solar'ом Designer'ом — знаменитым хакером и великим гуру, впервые применившим переполнение кучи для захвата управления машиной (на английском языке): http://www.openwall.com/advisories/OW-002-netscape-jpeg;

- **Once upon a free()…**
  - статья, опубликованная анонимным автором в PHRACK'e со ссылкой на рабоут Solar'a Desinger'a и детальным описанием сценария атаки под операционными системами Linux и System-V с упоминанием xBSD (на английском языке): http://www.phrack.org/issues.html?issue=57&id=9#article;
- **BSD Heap Smashing by BBP**:
  - презентация, прочитанная на Black Hate и описывающая сценарий атаки на phk-аллокатор, а точнее его незащищенную версию (на английском языке): http://www.blackhat.com/presentations/bh-europe-03/BBP/bh-europe-03-bbp.pdf http://www.blackhat.com/presentations/bh-europe-03/BBP/bbp-europe-03-tools.zip;
- **Exploiting OpenBSD by Ben Hawkes**:
  - презентация, прочитанная на RexCon и демонстрируя фундаментальные уязвимости в защищенном phk-аллокаторе на OpenBSD (на английском языке): http://www.slideshare.net/amiable_indian/exploiting-openbsd http://www.secguru.com/link/exploiting_openbsd_ppt;
- **The Malloc Maleficarum Glibc Malloc Exploitation Techniques**:
  - техника переполнения кучи в новых версиях glibc (на английском языке): www.derkeiler.com/Mailing-Lists/securityfocus/bugtraq/2005-10/0127.html www.derkeiler.com/pdf/Mailing-Lists/securityfocus/bugtraq/2005-10/0127.pdf;

>>> вставка

Аллаткатором (от англ. "to allocate" – размещать, выделять), называется совокупность механизмов, ответственная за распределение памяти и реализующая операции выделения, освобождения и (опционально) изменения размеров ранее выделенных блоков. Существует три типа аллокаторов: автоматические — работающие со стеком; статические - работающие с секцией данных и динамические — работающие с кучей. Если явно не оговорено обратное, то под аллокатором подразумевается именно динамический аллокатор.